VŠB – Technická univerzita Ostrava

Fakulta elektrotechniky a informatiky

Katedra informatiky

Spamové filtry založené na učení

Spam Filters Based on Learning

2013 **Pavel Sidžina**

VŠB - Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Zadání diplomové práce

Student: **Bc. Pavel Sidžina**

Studijní program: N2647 Informační a komunikační technologie

Studijní obor: 2612T025 Informatika a výpočetní technika

Téma: Spamové filtry založené na učení
Spam Filters Based on Learning

Zásady pro vypracování:

Vytvořte práci, v které budou popsány tři vybrané spamové filtry založené na učení. Práce bude obsahovat popis vybraných algoritmů a jejich srovnání na základě provedených experimentů.

Práce bude obsahovat tyto části:
1. Přehled a popis spamových filtrů založených na učení.
2. Analýza vybraných filtrů založených na učení.
3. Popis implementace a vlastní implementace vybraných filtrů.
4. Srovnaní vybraných filtrů na základě provedených experimentů.

Seznam doporučené odborné literatury:

[1] Ending Spam: Bayesian Content Filtering and the Art of Statistical Language Classification Jonathan A. Zdziarski ISBN:1593270526

Formální náležitosti a rozsah diplomové práce stanoví pokyny pro vypracování zveřejněné na webových stránkách fakulty.

Vedoucí diplomové práce: **Ing. Michal Prílepok**

Datum zadání: 16.11.2012
Datum odevzdání: 07.05.2013

doc. Dr. Ing. Eduard Sojka
vedoucí katedry

prof. RNDr. Václav Snášel, CSc.
děkan fakulty

Prohlašuji, že jsem tuto diplomovou práci vypracoval samostatně. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

Souhlasím se zveřejněním této bakalářské/diplomové práce dle požadavků čl. 26, odst. 9 Studijního a zkušebního řádu pro studium v bakalářských/magisterských programech VŠB-TU Ostrava.

V Ostravě dne 3.5.2013 .....

Zde bych chtěl poděkovat vedoucímu této práce, panu Michalovi Prílepokovi za trpělivost, konzultace při řešení problémů a poskytnutí dat k testování. Velké díky také patří mé přítelkyni, která mi pomohla tuto práci dát do správné gramatické formy.

ABSTRAKT

Cílem práce je popis a implementace 3 vybraných spamových filtrů, jejich testování a experimentování ve zpracování detekce. Vlastní vzorce filtrů zůstávají nezměněny, experimenty zkoumají jejich výsledky po změnách např. pareserování či dodatečných informací u slov. Díky těmto změnám lze vzorce srovnávat ve více úrovních a nalézt tak jejich klady a zápory či nejúčinnější úpravy.

KLÍČOVÁ SLOVA

Bayesovské spamové filtrování, Bayesovská kombinace, Robinsonův geometrický průměr, Fisher-Robinsonův inverzní chí-kvadrát, implementace kódů, testování efektivity, porovnání kódů, experimenty na kódech

ABSTRACT

The aim of this work is an implementation of 3 choosen spam filters, their testing and experiments in process of detection. Formulas stay the same, experiments explore their results after changing parser or words addition informations. Thanks to this changes, its able to compare formulas in more levels and find their positives and negatives or the most powerful editings.

KEY WORDS

Bayes spam filtering, Bayes combination, Robinson geometric mean test, Fisher-Robinson’s Inverse Chi-Square, implementation codes, testing efficiency, comparing codes, experiments on codes

SEZNAM ZKRATEK

Ham	- legitimní elektronická pošta
HTML	- značkovací jazyk pro hypertextové dokumenty
HTML Tag	- HTML značka označující část dokumentu
IP adresa	- číslo, které jednoznačně určuje síťové rozhranní
Spam	- nevyžádaná pošta

SEZNAM TABULEK

SEZNAM OBRÁZKŮ

SEZNAM GRAFŮ

1. Úvod

V dnešním kybernetickém světě, stojícím z velké částí na internetu a komunikaci přes něj, je téměř nemožné nenarazit na spam. Narazíme na něj všude, v elektronické poště, na sociálních sítích, v komentářích k článkům, v diskuzích, spamy také chodí po komunikátorech zvaných "instant messanger", zkrátka tam, kde je možné uživateli internetu podsunout reklamu či škodlivý kód.

Čistě spam však nedatuje svůj začátek s internetem. První jeho využití nalézáme již v 19. století, kdy se zprávy o pochybných investicích šířily telegrafem [1]. Opravdový vzestup však umožnily až novodobé technologie a hlavně veřejný internet, kde se emailová schránka stala největším bojištěm. Jelikož je nyní nevyžádané pošty tolik a zároveň je minimální šance potrestání za rozesílání, které je definováno zákonem 138/2002 Sb. o regulací reklamy, musíme se proti nim bránit sami. Postupem doby se vymýšlely techniky, jak proti nim bojovat, avšak tyto techniky nejsou stoprocentní a během chvíle někteří lidé zvládli nalézt jejich slabiny, a tak je obejít. Nyní se však většina technik stále používá jako podpora pro novodobé účinnější techniky a společně tvoří ještě silnější obranu. Mezi nejpoužívanější a nepříliš úspěšné patří whitelisting, kde uživatel zná adresy, ze kterých mu nikdy nechodí nevyžádaná pošta, naopak blacklisting využívá globální seznam emailových adres či IP adres, ze kterých nevyžádaná pošta chodí, a graylisting, který je dynamickou formou blacklistingu. Mezi ty úspěšné metody rozpoznávání obsahu či metody založené na učení.

Ve své diplomové práci se zabývám právě metodami založenými na učení. Metodika patří sice mezi pomalejší a robustní na data, fungující hlavně na individuálních datech, avšak její reálné využívání potvrzuje její velmi dobrou úspěšnost a použitelnost [2]. Ačkoliv název zní složitě, jedná se pouze o statistické vyhodnocování slov textu a následné využití statistiky k určení pravděpodobnosti, zda se jedná o spam, či nikoliv. Mezi mými vybranými kódy jsou 3 velmi odlišné vzorce. Každý používá jinou statistickou metodu na detekci, tudíž mají jiné charakteristiky. Jelikož tento vzorec je jen částí celého procesu, je v mých silách experimentovat v ostatních částech s daty a hledat tak lepší možnosti na zlepšení detekce u každého z nich. Důsledkem tak může být např. výrazné zlepšení detekce a překonání tak výsledků ostatních kódů.

V diplomové práci budu řešit využití těchto filtrů v e-mailech, avšak není zde velký rozdíl při využití např. v diskusích apod. Použití e-mailů má však jedno výrazné plus a to, že jsou na internetu dostupné reálné složky určené k testování.

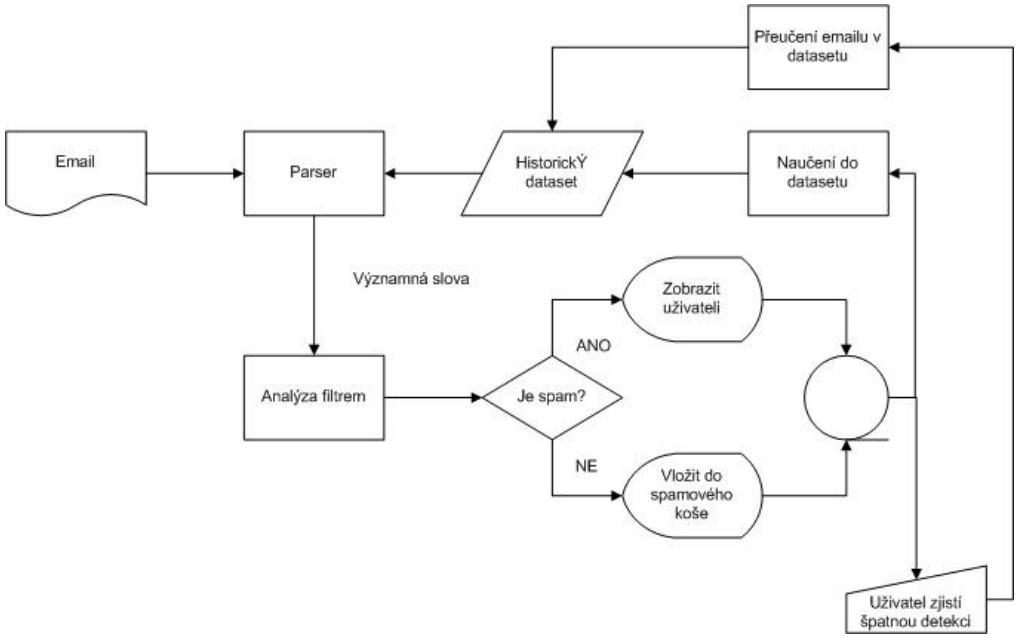
2. Popis spamových filtrů založených na učení

Způsobů, jak funguje spamový filtr založený na učení, je spoustu. Tyto varianty vznikají hlavně kvůli požadavkům, ať se jedná o rychlost, množství dat nebo uživatelskou individualitu, nicméně pouze duplikují jiné části, nebo je trošku ořezávají, smysl však zůstává stejný.

S touto kapitolou mi velmi pomohla kniha *Ending spam* [3], která mi byla velkou inspirací.

2.1. Struktura

Základní strukturu celého procesu můžeme vidět na *obrázku č. 1*.



Obrázek č. 1: Struktura procesu detekce emailů.

Email byl vložen do parseru, ten jej zpracuje a slova vyhodnotí s pomocí historického datasetu. Poté je předá části programu, která rozsoudí, zda se jedná o nevyžádanou poštu, či legitimní email. Jakmile tak učiní, email je znovu zpracován a jeho data přidána do historického datasetu na detekci pro další emaily. Uživatel však kontroluje výsledky tím, že nalezne spam mezi normální poštou, či ham ve spamovém koši. Jakmile usoudí, že došlo k omylu, označí tento email. Email se pak následně v datasetu přeučí na druhou variantu.

2.2. Historický dataset

Historický dataset je velmi důležitou součástí celého filtru. Uchovají se v něm veškeré informace o slovech, počtu zpracovaných emailů a jejich typech. Můžeme si to jednoduše představit jako databázi slov, obsahující informace o počtu výskytů a výslednou hodnotou slova jako na tabulce níže.

Slovo	Četnost ve spamech	Četnost v hamech	Hodnota
příklad	1	12	0,0625
je	5	5	0,45
…	…	…	…

Celkově spamů	Celkově hamů
250	200

Tabulka č. 1: Jednoduchá ukázka struktury datasetu.

Pokud začínáme tzv. "od nuly", historický dataset je prázdný, a přicházející emaily ho teprve začínají plnit. V takových případech bude chvíli trvat, než se filtr naučí a dosáhne své vynikající úspěšnosti. V jiném případě lze začít již s datasetem, který byl připraven jinde.

2.3. Učení datasetu

Existuje několik způsobů, jak se datasety mohou učit. Každý má své výhody i úskalí. My si ukážeme čtyři nejznámější. Jsou jimi: učení se ze všeho, učení se při chybě, ušení se dokud se neujistí a učení se dokud nechybuje [3].

2.3.1. Učení se ze všeho

Jako základní je učení se ze všeho. V tomto případě co email, to učení. Filtr tak reaguje rychleji na změny technik spamu, ovšem pokud je spamu v poměru k hamu příliš, začne se stávat, že hamy často označí za spamy. Stává se tak proto, že slova, která se sice vyskytují často v hamech, mají hodně záznamů ve spamech, ale v hamu ještě nepřišlo, či jen párkrát.

Další negativum nastává, pokud schránka přijímá mnoho emailů, jelikož po každém naučení emailu je třeba, aby se všechny hodnoty slov přepočetly. To může v tomto případě zabrat mnoho času.

2.3.2. Učení se při chybě

Učení při chybě řeší problémy učení se ze všeho, jelikož se učí pouze emaily, které špatně detekoval.V případě, že nám na email chodí mnohem více spamu, filtr jej pohodlně odhalí a znovu se tyto emaily neučí, čímž nebude snižovat hodnoty neutrálním a hamovým slovům. Dále nám to pomáhá při přepočítávání datasetu, kdy se nám sníží několikanásobně frekvence přepočtů.

Takovýto filtr však nezvládá reagovat na změny chování spamu, či změnu příchozích hamů. Jeho projevem je to, že takovéto emaily špatně detekuje a je na uživateli, aby je označoval programu, dokud se na ně nenaučí.

2.3.3. Učení se dokud se neujistí

Specifický typ učení, kdy se dataset učí pouze pokud email obsahuje slova, u kterých si není jistý, nebo došlo k chybě. Jedná se o střední cestu mezi prvními dvěma typy.

2.3.4. Učení se dokud nechybuje

Tento typ je odlišný od předchozích, nepracuje s emaily po přijetí, nýbrž jej spouští uživatel. Pokud se uživateli detekují emaily správně, pracuje bez něj. Jakmile však začne docházet k chybám, uživatel spustí přeučení. Program si zpracuje emaily a kompletně se naučí znovu. To však může zabrat v tyto chvíle mnoho času.

2.4. Parserování emailů

Email může obsahovat text, přílohy, html obsah, nebo také nic. Pro lepší pochopení funkce budu vysvětlovat vše na obyčejném emailu, který ničím nevyniká, tedy jde o klasický text *viz. ukázka č. 1*.

```
Ahoj Pavle, jsem rád, že si se ozval. Taky se mám skvěle, ale znáš to:
práce, škola - všechno to zabírá moc času. Už abych bydlel na Fidži! Mít
tak peníze. A co ty? Měj se!
```

Ukázka č. 1: Jednoduchý email pro názorné vysvětlení

2.4.1. Oddělovače

Pokud pohlédneme na uvedený text, hned nás napadne, jak bychom mohli získávat slova. Jedna z možností je rozdělovat pomocí znaků, které nepotřebujeme, např. mezera je nejvíce používaným znakem, který nám nic nepřinese. To nám však nestačí, jelikož je zde ještě několik znaků, které nejsou písmeny, a nemají pro nás význam. Kdybychom tedy podle této rady postupovali, u tohoto emailu bychom mohli říct, že množina oddělujících znaků (pro přehlednost pojmenujme jako O) by vypadala takto:

O = {**(mezera) , . : - ! ?**}

To nám však nemůže stačit, jelikož je zde mnoho dalších znaků, které se v jazyce používají a tvořily by nepříjemnosti ve slovech. Přidáme je tedy:

O = {(mezera) , . : - ! ? **_ ( ) ' "**}

Ale co když nám někdo pošle kus kódu napsaného v programovacím jazyce? Přidáme do naší množiny další oddělovače, které jsou častější, než se zdají:

O = {(mezera) , . : - ! ? _ ( ) ' " **[ ] { } < > * / + ` ; | \ @** }

To by byly základní znaky na oddělení, ale nesmíme zapomenout na znaky, které nejdou vidět, jako tabulátor, odřádkování apod. Naše množina se tedy opět rozroste:

O = {(mezera) , . : - ! ? _ ( ) ' " [ ] { } < > * / + ` ; | \ @ **\t \n \r \a \0** }

Bude toto opravdu stačit? Znaků, které dokážou oddělovat, či nemají žádný význam je velmi mnoho. Nabízí se tedy otázka, zda není lepší vyhledávat sekvence písmen. To je však vzhledem k množství jazyků a jejich rozpoložení v ASCII tabulce téměř nereálné. Vystačíme si tedy s uvedenou množinou, díky které zvládneme rozdělit velkou část textů na slova (i s čísly).

2.4.2. Zhodnocení nalezených slov

Pokud ilustrační email podrobíme rozdělení podle oddělovačů z množiny O, dostaneme tato slova (viz *tabulka č. 2*), ke kterým je přiřazena pravděpodobnost, což je hodnota z datasetu.

Můžeme vidět, že slova se zde neopakují, jelikož nám jde pouze o jeden výskyt slova v emailu. Každé slovo má zde svou pravděpodobnost, která je zde doplněna pouze ilustračně, avšak v reálném testu by vycházela z četností daného slova ve spamech a hamech v předchozích zpracovaných emailech a to podle *rovnice č. 1*., kde S je počet spamových emailů, kde se slovo vyskytlo, H je počet hamových emailů, kde se slovo vyskytlo, TS a TH jsou celkové počty spamových a hamových emailů, a P je výsledná pravděpodobnost, která se pohybuje v rozmezí $<0, 1>$.

$$P = \frac{\frac{S}{TS}}{\frac{H}{TH} + \frac{S}{TS}} \qquad (1)$$

Pokud by jsme vzali např. slovo "Fidži", které by mělo hodnotu $S = 1$, $H = 12$, a v té chvíli bychom měli zpracovaných *200 hamů* a *250 spamů*, dosazením čísel do *rovnice č. 1* dostaneme:

$$P = \frac{\frac{1}{250}}{\frac{12}{200} + \frac{1}{250}} = 0{,}0625$$

Výsledná hodnota nám pomáhá zjistit, s jakou pravděpodobností se slovo objevuje ve spamech či hamech. Hodnota 0,5 je neutrální hodnotou, směrem k 1 se slovo častěji objevuje ve spamech, směrem k 0 v hamech.

Slovo	Pravděpodobnost
ahoj	0,1
pavle	0,03
jsem	0,22
rád	0,35
že	0,53
si	0,43
se	0,51
ozval	0,19
taky	0,32
mám	0,16
skvěle	0,45
ale	0,49
znáš	0,21
to	0,41
práce	0,62
škola	0,22
všechno	0,39
moc	0,33
času	0,34
už	0,25
abych	0,19
bydlel	0,2
na	0,4
fidži	0,06
mít	0,41
tak	0,31
peníze	0,97
a	0,53
co	0,22
ty	0,15
měj	0,11

Tabulka č. 2: Slova emailu s jejich pravděpodobností.

2.4.3. Nejvíce významná slova

Jakmile máme přiřazeny hodnoty pravděpodobnosti ke slovům, můžeme nalézt nejvýznamnější slova. To, kolik těch slov bude, není přesně dáno, avšak většina metod, má své doporučené hodnoty,

či rozmezí [3]. Pokud slov bude mnoho, mohou nám výsledky vycházet příliš neutrální, pokud málo, může nám to u emailů, které obsahují významné spamové i hamové slova, špatně detekovat.

To, která slova vybrat, určuje (v případě počtu a ne rozmezí) vzdálenost od hranice, resp. vybereme počet slov, která jsou nejblíže hodnotám *0* a *1*. Alternativa je také, že vybereme počet slov, která jsou nejvíce vzdálená od *0,5*.

Jakmile tímto způsobem nelezneme nejvýznamnější slova (pro jednoduchost vybereme *5*) *viz tabulka č. 2* , můžeme se pustit do dalšího kroku.

Slovo	Pravděpodobnost	Vzdálenost
ahoj	0,1	0,4
pavle	0,03	0,47
jsem	0,22	0,28
rád	0,35	0,15
že	0,53	0,03
si	0,43	0,07
se	0,51	0,01
ozval	0,19	0,31
taky	0,32	0,18
mám	0,16	0,34
skvěle	0,45	0,05
ale	0,49	0,01
znáš	0,21	0,29
to	0,41	0,09
práce	0,62	0,12
škola	0,22	0,28
všechno	0,39	0,11
moc	0,33	0,17
času	0,34	0,16
už	0,25	0,25
abych	0,19	0,31
bydlel	0,2	0,3
na	0,4	0,1
fidži	0,06	0,44
mít	0,41	0,09
tak	0,31	0,19
peníze	0,97	0,47
a	0,53	0,03
co	0,22	0,28
ty	0,15	0,35
měj	0,11	0,39

Tabulka č. 3: Slova emailu s jejich pravděpodobností a vzdáleností od středu rozmezí. Slova vyznačená modrým pozadím jsou nejvýznamnější.

2.5. Hodnota emailu

Zde se dostáváme do fáze, kde se uplatňují různé statistické vzorce. Pravděpodobnosti nejvýznamnějších slov se zde vkládají do formulí, ze kterých dostaneme opět hodnotu, která však reprezentuje již celý email.

Pravděpodobnost se opět rozprostírá na ose v hodnotách od *0* do *1*, označující pravděpodobnost spamu či hamu. Blíže hodnotě *1* jsou spamy, k *0* hamy. Nicméně hranice, kde se rozhoduje, zda jde ještě o spam či ham, je pro každý vzorec jiná, jelikož výstupní hodnoty jsou po ose jinak rozprostřeny.

Pokud se naše výsledná pravděpodobnost bude nacházet mezi hodnotami *1* a hranicí spamu, můžeme označit email za spam, v opačném případě za ham.

2.6. Naučení se novému emailu

Po celkovém označení se dostáváme do fáze, kdy většinou email přidáme učícímu mechanismu. Funguje to velmi jednoduše. Všechny slova z emailu zapíšeme do naší databáze slov, pokud slovo v datasetu neexistuje, vytvoříme jej a zapíšeme si četnost u spamu či hamu (podle detekce), pokud existuje, inkrementujeme jeho četnost stejným způsobem u existujícího prvku.

Tato část se může mírně lišit podle zvoleného typu učení popsaného výše.

3. Analýza vybraných filtrů založených na učení

Mezi mé vybrané kódy patří Bayesovská kombinace, formovaná Paulem Grahamem, Robinsonův geometrický průměr Garyho Robinsona a Fisher-Robinsonův inversní chí-kvadrát opět Garyho Robinsona s využitím práce matematika Ronalda Fishera [3].

3.1. Statistické vzorce

Díky statistickým vzorcům můžeme s nějakou pravděpodobností predikovat, do jaké kategorie může spadat řešený email. Není divu, že jediný rozdíl v těchto filtrech nalezneme právě zde, jelikož každý z nich využívá jinou statistickou metodu. Výsledkem je pravděpodobnost, který se nalézá vždy mezi 0 a 1, kde blíže 0 je ham, blíže 1 spam.

3.1.1. Bayesovská kombinace

Bayesovská kombinace využívá tzv. Bayesovského teorému, který slouží pro výpočet podmíněné pravděpodobnosti[3][4]. Pro spamové filtry to využil Paul Graham a vzorcem viz. *rovnice č. 2.*, kde *P* je výsledná pravděpodobnost emailu, *A* je pravděpodobnost vybraného významného slova a *n* reprezentuje počet počítaných významných slov.

$$P = \frac{A_1 A_2 \dots A_n}{A_1 A_2 \dots A_n + (1 - A_1)(1 - A_2) \dots (1 - A_n)} \tag{2}$$

3.1.2. Robisnonův geometrický průměr

Bayesovská kombinace měří pravděpodobnosti na to, zda email je spam. Ovšem geometrický průměr Garyho Robinsona porovnává pravděpodobnosti na ham i spam a rozkládá emaily na ose výsledku rovnoměrněji [3]. Vzorce pro výpočet popisuje *rovnice č. 3.*, kde P je výsledná pravděpodobnost emailu, A je pravděpodobnost vybraného významného slova a n reprezentuje počet počítaných významných slov, S je výpočet spamové části, H je výpočet hamové části.

$$\begin{aligned} S &= 1 - \left((1 - A_1)(1 - A_2) \dots (1 - A_n) \right)^{\frac{1}{n}} \\ H &= 1 - \left(A_1 A_2 \dots A_n \right)^{\frac{1}{n}} \\ P &= \frac{1 + \left(\frac{S - H}{S + H} \right)}{2} \end{aligned} \tag{3}$$

3.1.3. Fisher-Robinsonův inversní chí-kvadrát

Další velmi odlišná technika je využití Gary Robinsonem inversního chí-kvadrátu matematika Ronalda Fishera, kde Gary Robinson našel lepší vlastnosti pro detekci než publikoval Paul Graham [5]. Výsledky stejně jako o předchozího Robinsonova propočtu rozkládá citlivěji emaily po výsledné ose. Vzorce pro výpočet popisuje *rovnice č. 4.*, kde *P* je výsledná pravděpodobnost emailu, *A* je pravděpodobnost vybraného významného slova a *n* reprezentuje počet počítaných významných slov, *S* je výpočet spamové části, *H* je výpočet hamové části a *Chí* je funkce počítající inversní chí-kvadrát.

$$H = Chí(-2ln(A_1 A_2 \dots A_n), 2n)$$

$$S = Chí\big(-2ln\big((1-A_1)(1-A_2)\dots(1-A_n)\big), 2n\big) \tag{4}$$

$$P = \frac{(1+H-S)}{2}$$

Jednoduchou funkci na inversní chí-kvadrát implementoval Tim Peters a později byla přejata i Gary Robinsonem pro jazyk C ve znění jaký je vidět v *ukázce č. 2* [3].

```
double chi2Q (double x, int v) {
  int i;
  double m, s, t;
  m = x / 2.0;
  s = exp(-m);
  t = s;
  for(i=1;i<(v/2);i++) {
    t *= m / i;
    s += t;
  }
  return (s < 1.0) ? s : 1.0;
}
```

Ukázka č. 2: Jednoduchý email pro názorné vysvětlení

3.2. Hranice spamu

Jak již bylo řečeno, všechny rovnice vrací hodnotu v rozmezí 0 až 1, kde blíže 1 je spam. Všechny mají svými tvůrci definovanou hranici, kde se email považuje za spam. Bayesovská kombinace má tuto hranici 0,9, což dává spamům poměrně malý prostor. Ve zbylých vzorcích je rozpoložení emailů na ose vyváženější a proto je hranice 0,55 [3].

3.3. Počet elementů

Oproti předchozí hranice spamu, kde je definována nejrozumnější hranice, počet elementů již závisí na implementaci. V Bayesovské kombinaci je doporučený počet 15 slov, Robinsonův průměr nemá definovaný počet a Chí-kvadrát má definované rozmezí hodnot slov, a odtud bere všechna, která splňují tuto podmínku a to, že hodnota musí ležet do 0,1 od kraje, tedy v intervalech <0, 0.1> nebo <0.9, 1> [3].

Jaký je nejlepší počet si můžeme odvodit. Pokud se budeme rozhodovat o celém emailu pomocí 3 slov, nemusí to být špatně, avšak pokud bude v nějakém hamu citován spam, či odesílatel napíše slovo, které se často používá ve spamech, máme velkou šanci, že bude detekce špatná. Pokud budeme navyšovat množství důležitých slov, může se nám stávat, že emaily začnou získávat neutrální hodnoty, jelikož budou ovlivněny mnoha lehce hamovými slovy. Je proto nejlepší najít rozumnou hodnotu či mez, aby slov nebylo až moc.

4. Popis implementace a vlastní implementace vybraných filtrů

Pro implementaci diplomové práce jsem si vybral jazyk C#, ve kterém se mi pracuje dobře a nejlépe se v něm orientuji. Jazyk má také mnoho dokumentace a je mnoho diskutován na fórech, což je nepatrnou výhodou oproti ostatním.

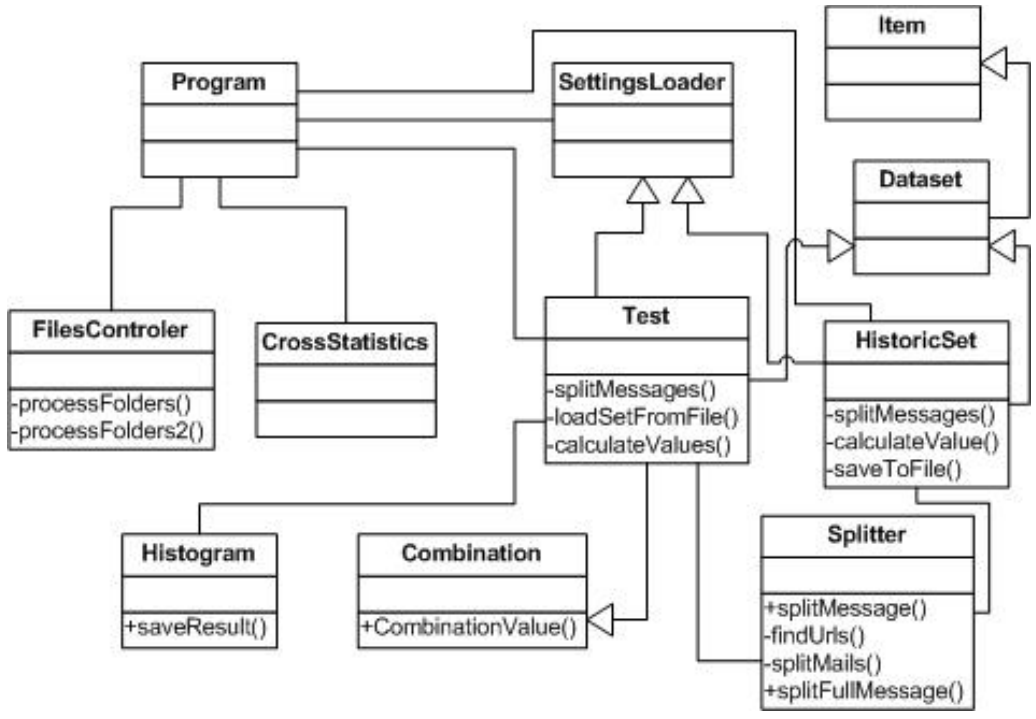
Měl jsem na výběr, jak implementovat dané kódy. Jedna varianta byla, že by každý kód měl svůj program. Od této varianty jsem však upustil, jelikož se kódy liší jen v jedné části postupu a každá úprava či oprava by znamenala provést ji 3x. Jelikož také plánuji testovat různé modifikace, opět se hodí, že je budu muset psát pouze jednou. Neposledním plusem je také samotné testování, kdy mohu spustit test, který ihned ozkouší dané nastavení na všech kódech.

Testování na souborech jsem musel provést v reprezentovatelné formě. Zprvu jsem vymyslel postup, že zadám cestu k složce souborů pro historický dataset a názvy složek pro testování. Jelikož bych ale takto mohl připravit testy tak, aby vyšly nádherně, přešel jsem na metodu zvanou křížový test. Jedná se desetinásobné testování. Vezmeme celý set emailů a rozdělíme jej rovnoměrně (spamy i hamy) do 10 složek a pro každou složku z ostatních vytvoříme historický dataset na kterém ji otestujeme.

Abych mohl porovnávat, vytvořil jsem několik možných výstupů, které program ukládá do csv souborů. Díky tomu je s nimi lepší manipulace, jelikož se dají otevřít v tabulkových programech či lehce zpracovat jinými programy.

4.1. Jedna aplikace, nespočet možností

Vybral jsem si tedy variantu, kdy existuje jeden program na všechno. Pakliže potřebuji mnoho funkcí, od prací se soubory na přípravu testování, nastavování testování, samotné testování, parserování, uchovávání datasetu, ukládání výsledků, aplikace se od prvotního návrhu velmi rozrostla. Její konečný návrh můžeme vidět na *obrázku č. 2*.



Obrázek č. 2: Třídní diagram aplikace

4.1.1. FilesControler

Jako první důležitá třída je *FilesControler*, která slouží k přípravě souborů pro křížový test. Pomocí cest se navede na složku se spamovými a hamovými emaily spočte si, kolik je souborů každého typu a rozdělí je na 10 částí.

Jelikož by se mohlo stát, že by rozdělení nemuselo být pro testování příznivé, vytvořil jsem 2 typy rozdělení do oněch 10 částí. První varianta je klasická, naplňuji složku dokud nenapočítám desetinu souborů. Jako druhou variantu jsem si zvolil rozdělení pomocí modula deseti, tudíž první soubor byl překopírován do první složky, druhý do druhé, jedenáctý opět do první apod. Tím si zaručuji již velmi náhodné rozdělení.

Soubory pro testování však mívají někdy v setech jména začínající "ham" či "spam" a jejich seřazení je poté nevyhovující pro testy, jelikož se nestává, aby nejdříve přišly všechny hamy a pak všechny spamy. Tyto soubory jsou při kopírování přejmenovány tak, aby se proházely spamy a hamy mezi sebou a také aby se promíchalo jejich pořadí. Proto je jejich nová forma xxxham.txt (popř. spam), kde *xxx* je náhodně generované číslo. Touto úpravou zaručuji, že soubory budou důkladně promíchané.

4.1.2. Dataset

Třída *Dataset* je třída, reprezentující databázi slov jako kolekci objektů třídy *Item* a přídružné informace nutné k výpočtům hodnot slov. Třída je také serializovatelná a je tedy ji možné ukládat a využívat vícekrát např. u testování více složek na stejném historickém datasetu.

4.1.3. SettingsLoader

Třída *SettingsLoader* slouží pro nastavení parametrů testování a cest k souborům. Tato třída obsahuje vše, co není dynamické a využívá se v programu. Pokud je potřeba něco z těchto hodnot změnit, změníme to jen zde, jelikož třída se načítá do všech částí programu.

4.1.4. Splitter

Třída *Splitter* přijímá emaily v různých podobách, podle různých nastavení se chová jinak. Obsahuje metody které vrací pole zpracovaných řetězců.

4.1.5. HistoricSet

Tato třída slouží pro vytvoření či načtení historického datasetu. Pokud set tvoříme, načteme zde všechny emaily ze složky nebo složek pro set a rozpaserujeme je na slova pomocí třídy *Splitter* a jejich metod. Poctivě se při plnění datasetu zapisuje zda slovo patří do hamu či spamu a také kolik spamů a hamů bylo zpracováno.

Jakmile je načtení hotovo, celý set se přepočítá a slovům jsou přiřazené hodnoty pravděpodobnosti hamu a spamu. Pokud budeme využívat tento historický dataset vícekrát, tak se v tuto chvíli uloží.

4.1.6. Combination

Třída *Combination* obsahuje statickou metodu *combinationValue*, která implementuje vzorce pro spočtení hodnoty emailu pro všechny vybrané kódy. Jaký kód spočte se určuje podle nastavení v *SettingLoaderu*. Přijímá pole významných slov a vrací double hodnotu emailu.

4.1.7. Test

Třída *Test* je zde pro testování složky popř. složek nad datasetem. Zde se načtou emaily na test a nechají se rozparserovat třídou *Splitter*. Mezi slovy se naleznou nejvýznamnější a pomocí nic se ve třídě *Combination* spočte hodnota emailu. Po výpočtu pomocí názvu emailu mohu zjistit, zda byla

detekce správná či ne, což se zaznamená do jednoduché třídy *Histogram*. Posléze se email podle nastavení naučí do datasetu se správným vyhodnocením.

4.1.8. Histogram a CrossStatistics

Histogram je třída, která během testování složky zaznamenává do kumulativních četností průběh špatných detekcí a ukládá jej do *csv* souboru pro další zpracování uživatelem.

Podobně jako třída *Histogram*, třída *CrossStatistics* slouží k záznamu průběhu testu. Nyní však zde sledujeme výsledky kompletního křížového testu a to počty chyb, úspěšnosti detekce a pod.

4.1.9. Program

Třída *Program* je spouštěcí třída. Vytváříme zde instance nastavení, historického setu, testu i křížových statistik.

4.2. Způsob uchovávání a vyhledávání dat

Data jsem se rozhodl uchovávat v paměti programu, jelikož na většinu testování to vystačí a je zde velký časový rozdíl zpracování oproti klasické databázi. Toto uchovávání má však omezení a to ve Windows, kde každý proces má přidělené maximum alokované paměti. Nicméně pro mé testování to vystačí.

Data ukládám do třídy *Dataset*, která se při implementaci měnila a to díky řešení rychlosti vyhledávání a vkládání dat. Jako první implementace byla vymyšlena tak, že slova byla uložena v kolekci *List<Item>*. *Item* je objekt obsahující slovo, četnosti v hamu a spamu a hodnotu slova. Aby však bylo možné rychle vyhledávat podle slova, bylo třeba mít list seřazen. Seřazování po každém vložení je však velmi neefektivní a proto jsem se odhodlal k implementaci vyhledávání půlením(vysvětlit a nakreslit zde).

Jelikož jsem zakládal dataset, mohl jsem *List* udržovat seřazen hned od začátku. Při vkládání jsem prohledával, zda slovo již neexistuje. Výhodou vyhledávání půlením je to, že mě zavede na místo, kde by prvek měl být. Nic tedy nebrání toho využít a nepoužitý prvek na této pozici vytvořit. Tímto způsobem mohu vkládat a vyhledávat v časové složitosti *log n*.

Oproti opakovanému seřazování o složitosti *n log n*, to byl velký posun. Nicméně jsem necítil, že by to bylo to pravé. Vedlo to k další úpravě, tentokrát k napodobení indexu databáze. Nyní jsem ponechal kolekci *List<Item>* avšak ji udržoval neseřazenou bez slova, pouze s hodnotami a k *Listu* přidal *Dictionary<string, int>*, kde klíčem je dané slovo a číselná hodnota je pořadí hodnot v *Listu*. Vkládání je tak konstantní, kdy se vloží hodnoty do *Listu* a následně slovo s pořadím do *Dictionary*. Vyhledávání poté funguje tak, že zjistíme, zda *Dictionary* obsahuje hledané slovo jako klíč, pokud ano, vrátíme si hodnotu pozice v *Listu*, pokud ne, můžeme provést vložení. Tato varianta funguje jako index a proto dosahuje oproti jiným řešením velké rychlosti zpracování.

Jelikož u experimentování se projevilo, že u jednoho testování by se hodilo mít i slovo přímo v listu, byl jsem nucen jej tam znovu redundantně vložit, což kupodivu moc nepřihoršilo alokované paměti.

4.3. Přidané možnosti

Jelikož je mou částí diplomové práce také experimentování, rozhodl jsem se implementovat i experimentální postupy. Díky implementaci v mém programu a testováním nad stejnými daty mohu reálně sledovat zhoršování či zlepšování detekce a vyhodnocovat jejich vlastnosti a účinky.

Nejen, že podrobím tyto experimenty testům na všech kódech, také mohu tyto funkce společně kombinovat a vytvářet tak různé varianty.

4.3.1. Počet významných prvků

Většina spamových filtrů má definovaný doporučený počet nebo rozsah prvků, podle kterých se filtr rozhoduje, zda je email spam či ham. Není tedy od věci do implementace přidat možnost změny počtu prvků pro každý test a zjistit, jak se mění charakteristiky rozložení emailů na dané sadě emailů. Takto můžeme zjistit, že se může více hodit jiný počet nebo rozsah prvků, než se používá.

4.3.2. Čísla jako oddělovače

Mezi další experimenty a funkce patří i možnost nastavit, aby se čísla braly jako oddělovače. Tedy změny původní množiny oddělovačů na:

O = {(mezera) , . : - ! ? _ ( ) ' " [ ] { } < > * / + ` ; | \ @ \t \n \r \a \0 **0 1 2 3 4 5 6 7 8 9**}

4.3.3. Velká a malá písmena

Máme dvě možnosti jak sledovat slova. Buďto můžeme například slovo "pískat" považovat za stejné se slovem "Pískat", které se liší pouze ve velikosti písmena, což se nám může stát na začátku věty. Slovo může být také jakkoliv napsané např. kapitálkami "PÍSKAT" či překlepem "píSKat". Zaručujeme takto, že slovo se vždy detekuje jako "pískat" a nebereme jej jako nové. Druhá možnost je brát tyto varianty slova jako různá slova. Pokud na to pohlédneme takto, nedává to příliš smysluplnosti, nicméně např. u slova "oko" a "OKO" se může jednat o název lidského orgánu ale také zkratka společnosti.

Díky této úvaze lze testovat, zda je lepší sledovat jen čistě slova, či i jejich variace s velkými písmeny, což díky této úpravě můžeme dělat.

4.3.4. Vykřičníky a otazníky

Dalším nápadem na experiment je odebrání vykřičníku a otazníku z množiny oddělovačů. Podobně jako velká a malá písmena se takto dosáhne více variant slov. Když pohlédneme na některé spamy, vidíme, že důležitá slova jsou často s mnoha vykřičníky, čehož by se mohlo využít. Množina tedy bude chudší o tyto znaky a bude vypadat následovně:

O = {(mezera) , . : - _ ( ) ' " [ ] { } < > * / + ` ; | \ @ \t \n \r \a \0 }

Pokud bychom kombinovali s čísly výsledná množina by vypadala takto:

O = {(mezera) , . : - _ ( ) ' " [ ] { } < > * / + ` ; | \ @ \t \n \r \a \0 0 1 2 3 4 5 6 7 8 9}

4.3.5. Sledování emailů a url adres

Jedná se o funkce, které jednoduše sleduje emaily a webové adresy užité v dodaném textu. Tyto části textu jsou pak ukládány jako samostatné jednotky s trochu jinými charakteristikami použitelnosti pro detekci. Jsou však uloženy jako obyčejná slova, která jsou doplněna o prefix spolu se nějakým znakem použitým v oddělovačích, čímž dosáhneme jednoduchého nalezení takovýchto slov v datasetu bez nutnosti přidávat další proměnnou. Výsledné slovo pro webovou adresu v datasetu poté může vypadat např. takto:

url**www.nazevstranky.cz*

Nicméně často se stává, že stránky či emaily nemusí být úplně stejné. Proto funkce sledování emailů a url adres disponuje ještě dělením daných adres na části.

Pokud tedy v emailu nalezneme url adresu www.subdomena.nazevstranky.cz/produkt/ a email emil@email.cz, do datasetu uložíme hned několik slov navíc a to:

*url**www.subdomena.nazevstranky.cz/produkt/*
*url-part**subdomena*
*url-part**nazevstranky*
*url-part**produkt*
*email**emil@email.cz*
*email-name**emil*
*email-domain**email*

Díky tomuto rozdělení můžeme lépe detekovat např. spamové emaily ze stejných domén, podsouvání stejných produktů, názvů stránek a subdomén.

4.3.6. Megaspam

Funkce megaspamu je mírně složitější, alespoň implementací. Sleduje totiž již při parserování, jestli slovo, které se v daném emailu objevilo již *n-krát*, nemá v datasetu spamovou hodnotu. Díky tomu můžeme přidávat na váze slovům, která jsou spamová a opakují se v emailu.

Jelikož se ale při hledání významných slov používá jen jeden výskyt slova, je třeba slovo modifikovat prefixem do doby, než se dostaneme v detekci do fáze, kdy hledáme ta nejvýznamnější slova v emailu. Poté opět musíme prefix odstranit, jelikož má pro nás význam pouze v části detekce a nesmí se nám dostat do datasetu, protože by se nám vytvořila nová modifikace slova.

Funkce má ale ještě další možnosti a to hranice, kdy se slovo začne považovat za megaspam a také, jak moc se přidá na váze slova při hledání těch nejvýznamnějších.

Nejlépe to uvidíme na jednoduchém příkladě *viz ukázka č. 3*, kdy budeme počítat s nastavením hranice megaspamu 2 a váze pro slovo 3. Počet významných slov pro jednoduchost budeme uvažovat 4. Když tedy dostaneme tento email, vidíme, že obsahuje 2x slovo "peníze".

```
Potřebujete peníze? Zavolejte 777777777. Dáme Vám peníze!
```

Ukázka č. 3: Jednoduchá ukázka emailu.

Za normální situace by významná slova byla vybrána jako v *tabulce č. 4*. Pokud pohlédneme na významná slova označena modře, vidíme, že se vybrala dvě spamová, jedno neutrální a jedno hamové.

Slovo	Hodnota	Vzdálenost
potřebujete	0,92	0,42
peníze	0,95	0,45
zavolejte	0,54	0,04
777777777	0,4	0,1
dáme	0,42	0,08
vám	0,22	0,28

Tabulka č. 4: Slova s jejich hodnotami a vzdálenosti v klasickém případě hledání významných slov, která jsou podbarvena modře.

Pokud však použijeme funkce megaspamu, naše výsledná tabulka bude vypadat jako levá část *tabulky č.5*, ale megaspamové slovo se nám zduplikuje a poté vybraná slova budou obsahovat 2 kopie slova "peníze" jako v pravé části tabulky, čímž je email jednoznačně určen za spam.

Slovo	Hodnota	Vzdálenost
potřebujete	0,92	0,42
mega*peníze	0,95	0,45
zavolejte	0,54	0,04
777777777	0,4	0,1
dáme	0,42	0,08
vám	0,22	0,28

Slovo	Hodnota	Vzdálenost
potřebujete	0,92	0,42
peníze	0,95	0,45
peníze	0,95	0,45
peníze	0,95	0,45
zavolejte	0,54	0,04
777777777	0,4	0,1
dáme	0,42	0,08
vám	0,22	0,28

Tabulka č. 5: Vlevo slova s jejich hodnotami a vzdáleností a megaspamem význačným prefixem "mega*", vpravo výsledek hledání významných slov podbarvený modře s kopiemi megaspamu.

4.3.7. Dvojce slov

Možnost sledovat i to, jak jsou slova za sebou naskládaná, je jistě zajímavá. Přináší to však i mnohem větší množství uložených dat, nicméně může to být velmi prospěšné, jelikož i stejná slova mohou nabývat více významů ve spojení s jinými slovy.

Daná spojení slov můžeme ukládat jako slova nová spojená libovolným oddělovačem, čímž zaručíme, že se spojení nebude moct nahradit jiným jedním slovem.

Další co tato možnost sledování nabízí je varianta, kdy můžeme sjednocovat slova podle pořadí jako např. na této ukázce:

$A * B \neq B * A$ a nebo $A * B = B * A$

4.3.8. Koeficient hamu

Když ve fázi tvorby datasetu nebo při učení nového emailu počítáme hodnotu slova, můžeme ji lehce modifikovat. Je to spíše individuální úprava, která by se mohla uplatnit při nerovnováze poměru spamů a hamů. Jedná se o modifikaci vzorce, kdy nadhodnotíme či podhodnotíme četnost u počtu nálezů u hamu jako v *rovnici č. 5*., kde k je diskutovaný koeficient ham, *S* je počet spamových emailů, kde se slovo vyskytlo, *H* je počet hamových emailů, kde se slovo vyskytlo, *TS* a *TH* jsou celkové počty spamových a hamových emailů, a *P* je výsledná pravděpodobnost, která se pohybuje v rozmezí *<0, 1>*.

$$P = \frac{\frac{S}{TS}}{\frac{k * H}{TH} + \frac{S}{TS}} \tag{5}$$

4.3.9. Sledování předmětu emailu

Stejně jako u sledování emailů a url adres obsažených v textu emailu můžeme, za předpokladu, že pracujeme s textem obsahujícím předmět emailu, sledovat slova obsažená v předmětu. Tato slova následně oddělit od klasického stejného slova opět jednoznačným prefixem a oddělovačem, čímž vytvoříme novou variantu slova.

```
Subject: Peníze na dosah
Potřebujete peníze? Zavolejte 777777777. Dáme Vám peníze!
```

Ukázka č. 4 : Jednoduchá ukázka emailu s předmětem emailu.

Pokud bychom dostali email, jako v *ukázce č. 4*, jeho rozkladem a využitím řešení předmětu by dopadlo jako v *tabulce č. 6*.

Slovo	Hodnota	Vzdálenost
potřebujete	0,92	0,42
peníze	0,95	0,45
zavolejte	0,54	0,04
777777777	0,4	0,1
dáme	0,42	0,08
vám	0,22	0,28
subject*peníze	0,99	0,49
subject*na	0,33	0,17
subject*dosah	0,42	0,08

Tabulka č. 6: Zpracovaný email s předmětem emailu.

Jak vidíme, slovo "peníze" zde máme 2x, avšak každý nám zastupuje jiné umístění a oba se mohou vybrat do významných slov nezávisle na sobě.

4.3.10. Rozsahový výběr významných slov

Metoda Fisher-Robinson používá pro významná slova o krajních hodnotách rozsahu. Kdežto zbylé 2 postupy používají maximální počet nejvýznamnějších slov. Proto program obsahuje také možnost, zapnout rozsahový výběr pro všechny filtry.

Fungování výběru můžeme pochopit z následujícího příkladu, kde opět použijeme jednoduchý vzor emailu z *ukázky č. 3*.

Jak vidíme na *tabulce č. 7*, levá strana obsahuje výběr pomocí počtu významných slov, pravá podle rozsahu. V tomto případě je rozdíl velmi patrný, jelikož do výsledku pravděpodobnosti nepromlouvají neutrální slova.

Slovo	Hodnota	Vzdálenost
potřebujete	0,92	0,42
peníze	0,95	0,45
zavolejte	0,54	0,04
777777777	0,4	0,1
dáme	0,42	0,08
vám	0,22	0,28

Slovo	Hodnota	Vzdálenost
potřebujete	0,92	0,42
peníze	0,95	0,45
zavolejte	0,54	0,04
777777777	0,4	0,1
dáme	0,42	0,08
vám	0,22	0,28

Tabulka č. 7: Vlevo klasické zpracování a vybraní významných slov počtem, vpravo vybraní významných slov rozsahem

4.3.11. Typ učení

Jak bylo napsáno v kapitole 2.3, můžeme implementovat několik typů učení. Proto jsem do aplikace implementoval možnost výběru učení, a to 2 nejzajímavější. Jako základ pokládám učení se ze všeho, jelikož je to podstatou celého filtru založeného na učení. Druhým stylem učení je učení se při chybě, který je časově ulehčená avšak méně flexibilní varianta.

Díky těmto dvěma variantám lze následně sledovat, jak moc a kde pomohou jiné experimenty a zda lze rychlejší variantu vylepšit kvalitativně do podobné úrovně.

4.4. Nastavení parametrů

Aplikace je vytvořena tak, aby testování na ni bylo pohodlné a hlavně efektivní. Pokud bychom chtěli otestovat experiment, který probíhá rychle, a množství testů by bylo větší, je lepší, aby se toto v programu nastavilo. Uživatel poté může test spustit a věnovat se jiným věcem, neboli nemusí každých pár minut nastavovat program jinak a spouštět jej znovu. Proto jsem vytvořil třídu *SettingsLoader*, kde je jeden z parametrů číslo verze, díky kterého tak můžeme pro každou verzi nastavit jiné parametry testování.Výsledné nastavení poté může vypadat např. pro testování na prvních 2 filtrech jako v *ukázce č. 5*.

I když se nastavení zdá zdlouhavé, většina parametrů zůstává stejná pro všechna testování, pokud se testuje na stejných a stejně umístěných datech. Proměnné, které se zde nastavují jsou několika typů. Cesty nastavují cesty na disku k souborům k testování a pro výstupní data. Ostatní je zapínání přidaných možností aplikace a popř. nastavení konzolového výstupu při zpracování. Jak vidíme, společné věci pro verze nemusíme psát ke každé verzi zvlášť, což je také velmi úsporné ve směru nastavování testování.

```
public SettingsLoader(int version, double edge = 0.0, int top = 0) {
            this.version = version;
            path_default = "D://TEST//";
            path_historic = "dataset-ver-" + version + ".data";
            path_output = "out//";
            path_histogram = "histogram";
            path_test = "version-" + version + "//";
            path_generated = path_default + "generated//";
            path_generated_linear = path_generated + "linear//";
            variant_learn = 1;
            type_generated = true;
             if (version == 1)
             {
                    versionName = "Graham ";
                    spam_edge = 0.9;
                    variant_combination = 1;
                    top_size = 15;
              }
              if (version == 2)
              {
                    versionName = "Robinson ";
                    spam_edge = 0.55;
                    variant_combination = 2;
                    top_size = 15;
              }
}
```

Ukázka č. 5: Ukázka nastavení parametrů v aplikaci pro testování

4.5. Možné výstupy aplikace

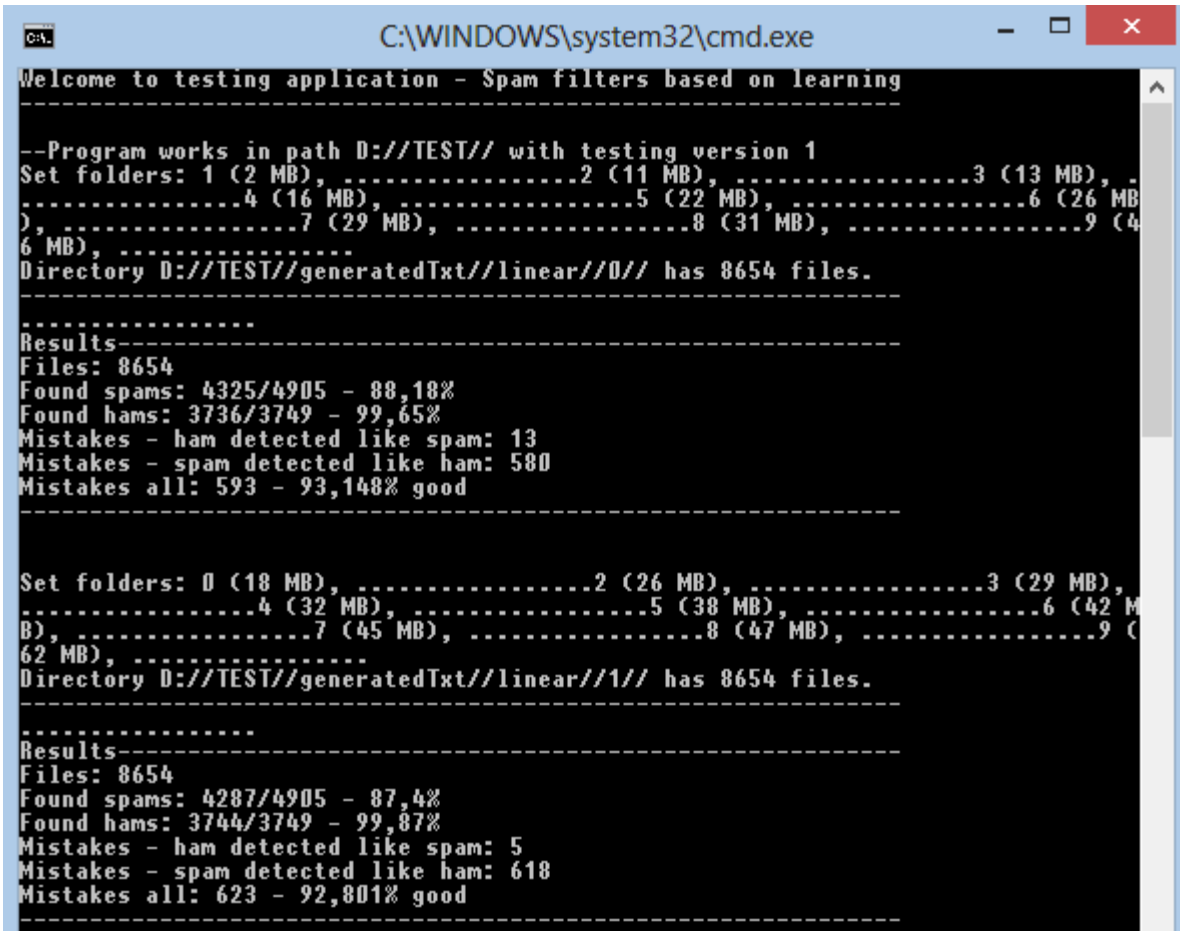
Aplikace má několik možných výstupů, některé jsou pouze informativní, některé důležité pro porovnávání.

4.5.1. Informativní výstup aplikace z konzole

Tento typ výstupu, jako je na *obrázku č. 3*, slouží pro informace při testování, můžeme tak sledovat, jak rychle probíhá zpracovávání emailů, zda program pracuje, jaké jsou výsledky testů a pod.

Na prvním řádku patřícího do testu vidíme v jaké složce pracujeme a jaká testovací verze je momentálně nastavena.

Další informace nám program dodává postupně. Jelikož pro testování využívám křížového testu, tak pro historický dataset je nutno zpracovat 9 složek (z 10 předgenerovaných) a na desáté otestujeme detekci. Zde se nám v progresu zobrazuje na jaké složce se v datasetu pracuje, vypíše se také množství alokované paměti, a program dále vypisuje tečky (podle nastavení) kolik emailů ze složky již zpracoval.



Obrázek č. 3: Informativní výstup aplikace z konzole

Jakmile projdeme 9 složek, spustí se nám testování, to poznáme vypsáním složky, na které se testuje a opět aplikace vypisuje tečky podle počtu zpracovaných emailů.

Po testu se nám na obrazovku vypíší výsledky:

Files - udává, kolik souborů bylo v testované složce
Found spams - zobrazuje kolik spamů bylo správně detekováno
Found hams - zobrazuje kolik hamů bylo správně detekováno
Mistakes - ham detected like spam - zobrazuje kolik hamů bylo detekováno jako spamy
Mistakes - spam detected like ham - zobrazuje kolik spamů bylo detekováno jako hamy
Mistakes - all - zobrazuje kolik bylo celkem chyb a celkovou přesnost celého testu

Toto je však pouze jedna smyčka z deseti v křížovém testu. Proto se dále rozjede nový test, který bude stejným způsobem vytvářet dataset. Rozdíl je v tom, že v datasetu vynechá jinou složku, kterou ještě netestoval a tu následně otestuje.

4.5.2. Průběh detekce

Výstup pro průběh detekce je typ výstupu, kdy je zaznamenán přesný průběh chyb detekce. Využití může mít na místech, kdy chceme porovnat např. rozdíl mezi detekcí klasickou a detekcí experimentální. Můžeme poté vidět, zda experiment pouze vylepšuje svůj úsudek, či celkově mění charakteristiku.

Zaznamenává se ve 3 rovinách: chyba detekce spamu, chyba detekce hamu a chyby celkem.

Aby se mohly výsledky porovnávat, musí se zapisovat každá detekce. Jak to funguje, vidíme na příkladě níže. Vrchní řada je seznam testovaných emailů, *H* značí ham, *S* značí spam. Spodní řada jsou výsledky, jak aplikace ilustrativně detekovala dané emaily. Spodní řada koresponduje pořadí řady horní.

S **S** S **H** H S **H** H S **S**
S **H** S **S** H S **S** H S **H**

Modře označeny jsou špatně detekované emaily. Již na prvý pohled vidíme, že celkově došlo k čtyřem chybám, dvě u spamu a dvě u hamu. Nicméně abychom z toho měli reprezentativní výstup pro všechny 3 vlastnosti, zapisujeme počet chyb po každé detekci a to pro každou vlastnost, i když ke změně nedošlo.

0 **1** 1 1 1 1 1 1 1 **2**
0 0 0 **1** 1 1 **2** 2 2 **2**
0 **1** 1 **2** 2 2 **3** 3 3 **4**

Toto je zápis, se kterým se již dá pracovat. Modře máme opět vyznačeny chyby. Aby bylo možné tento výstup lépe a lehce zpracovat, je ukládám třídou *Histogram* do csv souboru. Nejjednodušší využití těchto dat poskytují tabulkové programy.

4.5.3. Křížové statistiky a rozložení emailů na výsledné ose

Jako nejdůležitější výstup pro porovnání filtrů mezi sebou i pro vyhodnocování experimentů jsou křížové statistiky. Zde dostaneme všechny důležité data opět v jednom csv souboru pro jednu verzi křížového testu tzn. pro nastavení verze a 10 částí křížového testu.

Data, která lze využít k vyhodnocování a jsou obsaženy v souboru jsou:

Sum spam - počet spamů obsažených ve složce
Sum detected spam - počet spamů, které byly detekovány správně
Perc. of spam success - procento úspěšnosti detekce spamu
Sum ham - počet hamů obsažených ve složce
Sum detected ham - počet hamů, které byly detekovány správně
Perc. of ham success - procento úspěšnosti detekce hamu
Ham mistakes - počet špatně detekovaných hamů
Spam mistakes - počet špatně detekovaných spamů
All mistakes - celkový počet špatně detekovaných emailů
Perc. of success - procento celkové úspěšnosti detekce emailů
Time of testing - čas, jak dlouho test na složce trval (bez času tvoření datasetu)

Tato data jsou v jednom souboru pro všech 10 testů, resp. pro každou testovanou složku. Díky těmto datům můžeme pomocí dalších programů spočíst průměrné hodnoty a ty brát jako relevantní pro srovnávání pro různé testy, dále také srovnávat změny při experimentování jak na každé složce, tak na průměrných hodnotách.

Další část výstupu má jiné využití, a to sledování rozložení emailů na ose, resp. jejich vypočtená hodnota zasazena na osu v rozmezí *0* až *1*. Tato data jsou ze všech testů v křížovém testu a jsou sledovány ve třech rovinách pro ham i spam, i když je využita reálně jen jedna ze tří, zbylé slouží pouze pro kontrolu či zvědavost po přesných datech. Všechny hodnoty jsou děleny do 101 polí. Prvních 100 je rozmezí od *0* do *1* po jedné setině, tedy <0; 0,01), <0,01; 0,02) ... <0,99; 1.00). Posledním polem jsou emaily, které nedostaly žádné ohodnocení, jelikož neobsahovaly ani jedno významné slovo. Kdy tomu nastane a jak se tento stav řeší, je vysvětleno dále v *kapitole 4.6.3*.

Datovány jsou tyto veličiny:

Ham - počet hamů obsažených ve složce
Total ham - je kumulativní hodnota, kolik hamů se celkem umístilo od prvního pole do tohoto pole, jednoduše řečeno, pokud v první poli byly 2 výskyty a v druhém 6, v absolutních hodnotách bude v prvním poli 2 výskyty, v druhém již 8 (2 výskyty z předchozích polí + 6 vlastních)
Total ham perc. - je veličina propočtena z *Total ham*, kdy známe celkový počet hamů a můžeme tedy spočíst, kolik procent z celkového počtu hamů se nachází mezi nulou a určitým polem.

Stejné veličiny jsou datovány také pro spam.

4.6. Komplikace při implementaci

Během implementace jsem narazil na nemálo problémů. Mezi ty malé a nepříliš významné patřily problémy s rychlostí zpracování, které lze lehce a rychle vyřešit. Většina těch větších se však týkala logiky fungování spamového filtru, které nebývají nikde moc popsány.

4.6.1. Malé emaily

První větší problém nastal ve chvíli, kdy jsem pracoval pouze s čistým textem emailu, tedy bez ostatních položek jako předmět, od koho přišel, komu všemu byl hromadně odeslán a také bez html značek a podobně.

Za takové situace se mohlo velmi jednoduše stát, že došlý email nakonec neobsahoval žádné slovo a tedy nebylo z čeho počítat hodnotu emailu, ani vydedukovat, zda se jedná o ham nebo spam.

Problém jsem musel vyřešit tak, že jsem odstranil z testování emaily, obsahující 1 slovo a méně. Teoreticky jsem mohl odstranit emaily např. do deseti slov, avšak i takové malé emaily mohou mít velkou váhu ne při rozhodování o své detekci, ale při následném rozhodování u ostatních emailů. Proč je tomu tak pochopíme u dalšího problému nazván *Hapaxy*.

4.6.2. Hapaxy

Po implementaci prvního filtru jsem otestoval, jak na tom s detekcí je. Nicméně nedošel jsem daleko, jelikož aplikace brzy skončila s chybou dělení nulou. Rozhodl jsem se prozkoumat, zda jsem při implementaci neudělal chybu. Chyba se však nalezla až při debuggování (neboli procházení hodnot proměnných za běhu programu), kde jsem narazil na problém dělení nulou u výpočtu hodnoty emailu.

Pokud pohlédneme znovu na vzorec tvorby hodnoty slova (*viz. rovnice č. 1*), po dosazení hodnot zjistíme, že při stavu, kdy email nemá žádný spamový výskyt, slovo dostane hodnotu 0. Pokud nemá hamový výskyt, hodnota je 1, viz. výpočet níže.

$$P = \frac{\frac{S}{TS}}{\frac{H}{TH} + \frac{S}{TS}}$$

$$P = \frac{\frac{0}{TS}}{\frac{H}{TH} + \frac{0}{TS}} = \frac{0}{\frac{H}{TH}} = 0$$

$$P = \frac{\frac{S}{TS}}{\frac{0}{TH} + \frac{S}{TS}} = \frac{\frac{S}{TS}}{\frac{S}{TS}} = 1$$

Ani jedna z hodnot není špatně. Pokud však takové extrémní hodnoty dosadíme do vzorce níže, který je pro Bayesovskou kombinaci, zjistíme, že stačí jedno takové slovo a email je automaticky detekován jako nejjistější spam nebo ham.

$$P = \frac{A_1 A_2 \dots A_n}{A_1 A_2 \dots A_n + (1 - A_1)(1 - A_2) \dots (1 - A_n)}$$

$$P = \frac{0 A_2 \dots A_n}{0 A_2 \dots A_n + (1 - 0)(1 - A_2) \dots (1 - A_n)} = \frac{0}{(1 - A_2) \dots (1 - A_n)} = 0$$

$$P = \frac{1 A_2 \dots A_n}{1 A_2 \dots A_n + (1 - 1)(1 - A_2) \dots (1 - A_n)} = \frac{A_2 \dots A_n}{A_2 \dots A_n} = 1$$

Pokud se objeví taková slova dva, každé z jedné strany osy, program skončí na chybě při pokusu dělení nulou.

$$P = \frac{01 A_3 \dots A_n}{01 A_3 \dots A_n + (1 - 0)(1 - 1)(1 - A_3) \dots (1 - A_n)} = \frac{0}{0} = Chyba$$

Jakožto první řešení jsem zamezil těmto slovům dostat se mezi významná slova. Aplikace teď již neskončila programovým neúspěchem, avšak výsledky úspěšnosti detekce se pohybovaly kolem 40%. Logicky není totiž správné tato slova vypouštět, jelikož tak vypustíme i slova, které mohou být nalezená tisíckrát ve spamu a nikdy v hamu. Bez těchto slov přicházíme o důležitá slova pro detekci. Dalším krokem bylo vyhledat, jak taková slova řešit.

Řešení jsem nalezl v knize Ending Spam [3], a to tak, že těmto slovům nastavíme hodnotu buď 0,99 pokud jde o spam, nebo 0,01 pokud jde o ham. Tímto zaručíme, že ve vzorcích nenastane ani jedna z těchto chyb.

Jakmile jsem provedl zmíněnou úpravu, ihned jsem otestoval, zda to pomohlo a jestli to již bude opravdu dobře. Jelikož jsem pracoval pouze s textem z emailu, nečekal jsem extrémní výsledky i když je známo, že úspěšnost spamových filtrů založených na učení se nachází kolem 99%(asi odkaz na něco?). Po testování se moje úspěšnost pohybovala mezi 60 - 90%, což mě vedlo k tomu, že ještě není vše v pořádku.

Proto jsem se uchýlil znovu do četby většího bloku, obsahujícího i předchozí radu, nazvaného "Hapaxes", což by se volně dalo přeložit jako "něco co bylo řečeno pouze jednou". Celý odstavec pojednával o této problematice zahrnující i problém s příliš určitými slovy.

Jakmile se dostaneme do stavu, kdy slovo má jednu detekci hamu a žádnou spamu, slovo dostane hodnotu 0,01. Nicméně i na výsledcích bylo znatelné, že slova, která mají malou četnost, by neměly dostat možnost rozhodovat o hodnotě emailu. Proto se zavádí mez použitelnosti, kde slovo nemající dostatečně dost četností dostane buď neutrální hodnotu, nebo se nepoužije vůbec.

Pravidlo, které jsem nakonec použil bylo, že slovo musí mít buď alespoň 7 četností, nebo 6, avšak za podmínky, že z daných emailů bude ham jen jeden, nebo více než 2. Po použití se výsledky při použití pouze textů z emailu dostávalo vždy nad 90%.

4.6.3. Emaily bez významných slov

Významná slova rozhodují o hodnotě emailu, avšak jakmile nemáme žádná, nevěděl jsem, na jakém základě se rozhodnout. Důležité je zjistit, kdy se takto stane.

První možnost je, když pracujeme pouze s textem emailu a přijde prázdný text emailu. Daný email může obsahovat přílohu, která autorovi stačila pro odeslání, nebo bylo vše důležité řečeno v předmětu. V tomto případě nelze nijak rozhodnout zda jde o spam či ham, protože takové emaily mohou chodit z více stran, avšak nikdo by si nepřál aby např. poslané fotky v příloze spadly do spamového koše. Je to nebezpečné, avšak takovéto emaily by měly být označené jako hamy.

Další možnost je, že přišel email např. s pár slovy, ale žádné z nich dataset neznal. Řešení je stejné jako předchozí: neměli bychom označovat takovéto emaily za spamy.

Poslední možnost je, že slova sice zná, jedná se o již zmíněné hapaxy. Jak bylo v minulé podkapitole řečeno, tato slova by měly mít buď neutrální hodnotu, či se nebrat za slova významná. Z tohoto důvodu je třeba i v tomto případě email označit jako ham.

Z mého pohledu i pro statistiky bylo důležité si toto uvědomit a také tyto emaily škatulkovat do samostatného výsledku abych mohl vidět, jak často se mi to stává.

4.6.4. Kompletní emaily

Kompletní emaily jsou emaily z pohledu zdrojového kódu tzn. s kompletním popisem všech částí a s tagy. Ačkoliv email může přijít bez textu a předmětu i tak může obsahovat velké množství informací. To jsou jejich výhody avšak i zápory.

Spousta informací sebou nese i nestandardně uspořádaní. Není totiž definováno v jakém pořadí se mají informace zapisovat, nemusí být všechny uvedeny a nejsou sepsány v lehce zpracovatelné formě jako např. xml. Parserování informací z tolika možných variant zápisu je posléze velmi obtížná věc.

Jelikož jsem chtěl jen některá data jako předmět, odesílatel, příjemci, kopie, skryté kopie a text emailu v jeho formě i s tagy, musel jsem vytipovat, jak tato data nalézt a jak nalézt jejich ukončení. Nakonec jsem musel vyhledávat přímo slova definující tyto části, sledovat zda již se nejedná o text z jiné části a hledat ukončení části jež se používá "\n", což je zápis pro ukončení řádku. Avšak nastal zde problém, kde při kopiích emailu s více emaily jsou i tyto emaily odděleny zalomením řádku. Nejtěžší však bylo vyhledat text emailu, který zde nemusel ani být. Intuitivně jsem však vytipoval, že by se měl nacházet za dvojitým zalomením řádku.

Z těchto důvodů jsem vytvořil pouze jednoduchý parser emailů pro, dle mého, nejdůležitější části.

Problém to však není jediný. Další nastává s větším množstvím emailů. Pokud jsem se pokusil zpracovat 80 tisíc emailů bez jakéhokoliv pravidla, tedy zpracoval úplně vše, již ve 3/4 zpracování jsem dosáhl hranice maximální možné hranice alokování paměti pro spuštěný proces ve Windows. Pokud jsem zpracovával pouze data, která jsem chtěl, problém nastal při zpracovávaní smyček křížového testu.

Jelikož bylo toto velmi náročné na čas, než jsem se dostal k danému přetečení a nejsem si jist, jestli by šlo toto nějak lehce vyřešit, rozhodl jsem se tuto část testování přenechat pouze pro testování na menším setu emailů.

4.6.5. Stop slova

Stop slova jsou slova, která nenesou ve větě žádný význam. Pro češtinu jsou taková slova např. spojky, předložky, v angličtině i "a", "the" apod. [6]. Čistě teoreticky by nám takovéto slova neměly ničím pomoct při detekci spamu a hamu, když nenesou žádnou informační hodnotu.

Ovšem zde nastává problém, jelikož každý jazyk má většinou jiná stop slova. Ačkoliv některá zde nenesou význam, stejně psané slovo v jednom jazyce může v jiném jazyce význam mít. Tudíž z toho vyplývá, že pokud bychom chtěli využít odebírání bezvýznamných slov, museli bychom pracovat nad emaily, které nejsou v mnoha jazycích, a pro každý z nich do aplikace vložili dané slova.

Jelikož mé testování probíhá nad sety emailů, které jsou v mnoha jazycích, slova neodebírám a nechávám je pomoct rozhodovat o hodnotě. V jednom experimentu sleduji také dvojce sousedících slov, kde i tyto bezvýznamné slova spojením význam mohou nabýt. Pokud se objeví takové slovo v historickém datasetu, bude mít pravděpodobně neutrální hodnotu, jelikož se např. spojky a předložky vyskytují jak ve spamech, tak i hamech.

5. Srovnaní vybraných filtrů mezi sebou a na základě provedených experimentů.

V této kapitole se budu věnovat reálnému testování daných spamových filtrů založených na učení, jejich otestování a uplatnění experimentů na ně. Testy budou probíhat na dvou setech emailů a budu se snažit vylepšit hodnoty výsledků na každém filtru. Mohu poté takto porovnávat filtry mezi sebou, experimenty na různých filtrech, a celkové maximální vylepšení detekce filtru.

5.1. Využité testovací sady

Pro své testování jsem využil dvě testovací sady: Trec05 public corpus a SpamAssassin corpus, jejichž podrobnosti nalezneme v *tabulce č. 8*.

Název	Počet emailů celkem	Počet spamů	Počet hamů	Poměr spam/ham
TREC05	92 189	52 790	39 399	1,34
SpamAssassin	16 298	9 348	6 950	1,34

Tabulka č. 8: Využité testovací emailové sady

5.1.1. Sada TREC05

Tato sada byla uveřejněna v roce 2005 v souvislosti s konferencí "Text REtrieval" známou jako TREC [7]. Charakteristiky sady můžeme vidět v *tabulce č. 8*.

Jedná se o velmi obsáhlou sadu, na které díky tomu lze vidět lépe, jak rychle se filtry učí a jak zabírají experimenty. Pokud bychom pracovali s příliš malou sadou, výsledky by takto mohly být zkreslené. Sada je také ideální pro testování jiné verze učícího procesu, protože např. u učení se jen při chybě dojde stále k velkému počtu učení a můžeme tak pozorovat reakce filtru po větším množství chyb.

Sadu mám ve dvou formách. První jsou klasické emaily, druhá obsahuje pouze vyexportované texty z těla emailu. Pro testování jsem využil druhou formu, jelikož první obsahuje přespříliš informací, i po odstranění nedůležitých částí, což vede k velké časové i paměťové neefektivitě při testování a špatné použitelnosti.

5.1.2. Sada SpamAssassin

Tato sada je vytvořena kolektivem pracujícím na filtračním programu SpamAssassin. Jedná se o veřejný soubor emailů určen pro testování filtrů [8]. Jedná se o několikrát menší sadu než sada TREC05.

Hodí se velmi na testování, jelikož půjde vidět, jak filtry pracují na menším souboru dat a jakých účinků dosáhnou experimenty. Sada je opět ve dvou formách jako předchozí, nyní však využiji obě, jelikož zde nebude tak velká náročnost na čas a paměť.

5.2. Způsoby vyhodnocování testování

Nejdůležitější částí celé diplomové práce je srovnání filtrů mezi sebou a experimentování, kde je potřeba také srovnávat, zda filtru pomohly či ne. Proto je třeba najít metodu, jak správně rozhodnout, co je lepší.

V našem případě máme dvě množiny, spamy a hamy. Pokud je vložíme do spamového filtru, dostaneme jako výstup opět dvě množiny a to: email detekován jako spam a email detekován jako ham. Jelikož se nestává, že by filtr vždy stoprocentně klasifikoval vše správně, výsledkem jsou pro nás 4 množiny které vidíme na *tabulce č. 8*.

		Reálný typ	
		spam	*ham*
Detekovaný typ filtrem	*spam*	TP	FP
	ham	FN	TN

Tabulka č. 9: 4 možné množiny po klasifikaci filtrem, T = true (pravdivý), F = false (nepravdivý), P = positivní nález (spam), N = negativní nález (ham).

Pokud náš filtr nalezne email, o němž tvrdí že je spam, označíme jej jako pozitivní nález, pokud ham tak negativní nález. Když poté porovnáme výsledky filtru s realitou, označíme zda detekoval správně nebo ne. Pro klasifikaci správnosti používáme anglické true a false (pravdivý a nepravdivý).

5.2.1. Porovnání pomocí přesnosti

Přesnost je veličina udávající úspěšnost pomocí *rovnice č. 6*., kde *FP* a *FN* jsou chyby detekce podle *tabulky č. 9, P* je reálný počet spamů, *N* je reálný počet hamů. Jejich sečtením dostaneme v děliteli vzorce celkový počet testovaných emailů. Výsledkem je reálné číslo v rozmezí *(0 , 1)*.

$$Přenost\ (Acc)\ = 1 - \frac{FP + FN}{P + N} \tag{6}$$

Pokud tuto přesnost vynásobíme 100, získáme procento správně detekovaných emailů.

Otázkou zůstává, jak moc je toto dobré k porovnání složitějšího problému jako zde. Pokud totiž pohlédneme na vzorec, můžeme vidět, že zde vůbec nezáleží na tom, v jaké veličině špatná detekce nastala.

Pro příklad: za situace, kdy máme 1000 emailů, 500 spamů a 500 spamů, a na výstupu filtru bude FP = 10 a FN = 100, dostaneme pomocí výpočtu níže výsledek 0,89, což je ve výsledku úspěšnost 89%. Nicméně pokud za stejné situace budeme mít FP = 100 a FN = 10, výsledek bude stejný, avšak pro nás to bude znamenat, že se v poště objeví sice jen 10 spamů ale 100 hamů zahodí. Toto je neakceptovatelné a proto pouze výsledek přesnosti nemůže rozhodovat o efektivitě filtru.

$$Přenost\ (Acc)\ = 1 - \frac{FP + FN}{P + N} = 1 - \frac{10 + 100}{500 + 500} = 0{,}89$$

Jedna z možností jak tyto hodnoty využít je brát ohled na počet špatně detekovaných hamů a počítat s možnou neúplnou přesnosti. Pro jednoduché porovnání, kde nedochází k zásadním změnám podobným jako v příkladě, či jde o sledování změn informativní jako při experimentech, tyto hodnoty postačí.

5.2.2. Porovnání pomocí vážené přesnosti

Pro případ, kdy chceme, aby se výsledek přesnosti odvíjel od toho co potřebujeme, můžeme využít váženou přesnost. *Rovnice č. 6* se tak zkomplikuje, viz *rovnice č. 7*, kde *TP, FP, FN* a *TN* jsou stavy detekce podle *tabulky č. 9, P* je reálný počet spamů, *N* je reálný počet hamů. Jejich sečtením dostaneme v děliteli vzorce celkový počet testovaných emailů. Výsledkem je reálné číslo v rozmezí *(0 , 1)*.

$$Vážená\ přesnost\ (WAcc) = \frac{\lambda * \mathrm{TN} + \mathrm{TP}}{\lambda * N + P}$$
$$\text{kde } \lambda = \frac{\mathrm{uFP}}{\mathrm{uFN}} \tag{7}$$

Jak vidíme, lambda vyjadřuje poměr mezi špatně detekovanými hamy k špatně detekovaným spamům. Prefix *u* je zde proto, že si jakožto uživatelé určíme, jaký tento poměr má být. Jedná se o číslo $\in (0, \infty)$. Pokud např. zvolíme, že jsme schopni riskovat zahození jednoho hamu s tím, že za to necháme zobrazit 100 spamů, naše lambda bude 0,01. Tímto poměrem poté vynásobíme celkový počet spamů a správně detekovaných spamů.

Po aplikaci na orientační příklad výše, kde FP = 10 a FN = 100, dostaneme úspěšnost 98%.

$$\lambda = \frac{\mathrm{FP}}{\mathrm{FN}} = \frac{1}{100} = 0{,}01$$

$$Vážená\ přesnost\ (WAcc) = \frac{0{,}01 * 400 + 490}{0{,}01 * 500 + 500} = \frac{494}{505} = 0{,}98$$

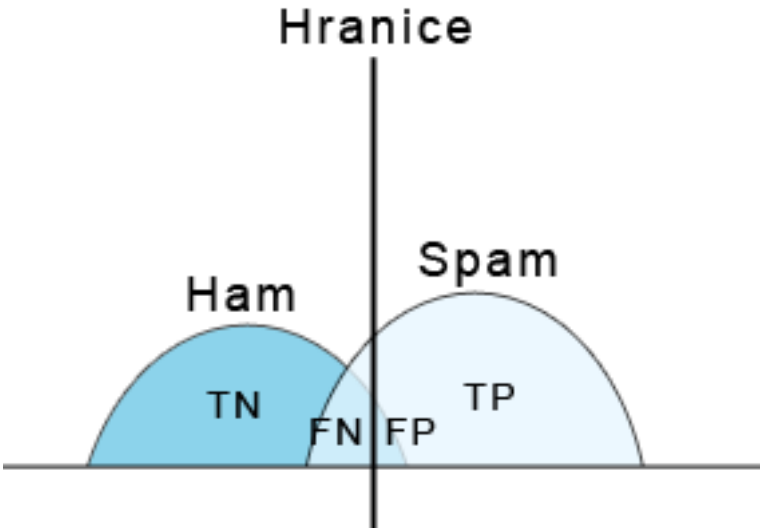
Pokud bude příklad opačný, tedy FP = 100 a FN = 10, výsledek dosáhne procentuální úspěšnosti pouze 80%.

$$Vážená\ přesnost\ (WAcc) = \frac{0{,}01 * 490 + 400}{0{,}01 * 500 + 500} = \frac{404{,}9}{505} = 0{,}8$$

5.2.3. Porovnání pomocí AUC

Pokud však půjdeme do hloubky problému, narazíme na metodu pro srovnání AUC (anglicky *area under curve*, česky *oblast pod křivkou*). Nejdříve se však musíme vrátit trošku zpět.

Jak bylo ukázáno na *tabulce č. 9*, po klasifikaci spamovým filtrem získáváme 4 množiny emailů. Proč tomu tak je nám pomůže osvětlit obrázek.



Obrázek č. 4: Ilutrace problému s hranicí spamu

Na *obrázku č. 4* vidíme větším písmem 3 důležité části. Množina ham, jež reprezentuje reálné hamy a jejich rozložení na ose po určení hodnoty filtrem. Spam zase reprezentuje spamy. Hranice znázorňuje hodnotu, podle které filtr rozhoduje, že jde o spam či ham. Pokud by nastal ideální případ, množiny se neprotínají a tím pádem by stačilo hranici vložit mezi ně a dostaneme od filtru dokonalý 100% autentický výsledek.

Bohužel tomu tak nebývá a proto musíme hranici umístit tak, abychom dosáhli nejlepší efektivity. Pokud někde vložíme hranici, vytvoří se naše již známé oblasti. TN můžeme vidět tmavší modrou do hranice. Chyba špatně identifikovaných hamů, tedy FP, je zbytkem tmavě modré za hranící. Stejně tak TP je světle modrá do hranice, zbytek nalézající se za hranicí je FP. Nicméně opět potřebujeme, aby FP bylo mnohem méně než FN, proto by hranice měla mířit více ke konci hamové množiny.

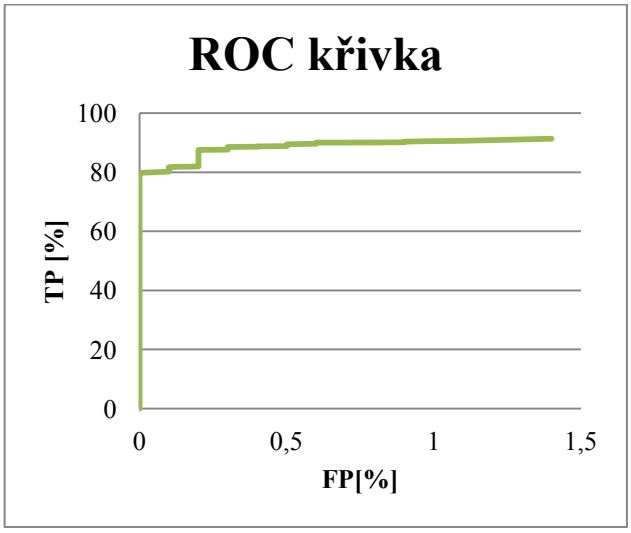
To, kde hranici umístit a posléze spočíst AUC, nám pomůže křivka ROC. Ta byla původně používaná pro oddělení šumu od signálu. Nyní její využití zasahuje do mnoho oblastí, kde našla své uplatnění např. i v medicíně [9].

Křivku ROC můžeme vytvořit několika způsoby a to poměrem (osa x ku osa y) TP ku FP, 1-(TN/(TN+FP)) ku TP/(TP+FN) apod. [10]. Výsledkem je vždy stejná křivka, pouze se rozdílnými osami. Křivku můžeme vidět na g*rafu č. 8*.

Pro nejlepší nastavení hranice, neboli prahu, existuje mnoho metod. Některé obsahují dodatečné váhy kvůli více proměnných a individuálních potřeb, nicméně pro jednodušší vysvětlení většinou stačí nalézt bod na křivce, který je nejblíže levému hornímu rohu [10]. Pokud zvolíme tento bod jako práh, měl by náš filtr dosahovat optimálního poměru chyb spamu a hamu.

Ovšem jelikož my již máme definované nejlepší prahy od samotných tvůrců filtrů, stačí se nám zabývat AUC. Pokud opět pohlédneme na *grafu č. 1*, náš hledaný AUC je plocha, která se nalézá pod ROC křivkou [10]. Pokud chceme porovnávat filtry pomocí AUC, musíme spočíst kolik procent z celkové plochy zabírá. Pokud filtr zabírá více plochy, je ideálnější [10].

Často se porovnávání interpretuje jako (1 - ROCA)[%], což je zbytek po odstranění AUC a nejmenší výsledek je brán jako nejlepší.



Graf č. 1: ROC křivka

Díky této metodě můžeme poměrně dobře porovnávat filtry. Pokud budeme pohlížet na 2 různé grafy, rozdíly v efektivitě by měly jít většinou vidět na první pohled. Důležité však je, že i když filtr bude mít lepší výsledky zabrané plochy, neznamená to, že výsledky testu musí být nutně lepší výsledek testování, jelikož tyto hodnoty vypoví hlavně o celkovém výkonu filtru [4].

5.2.4. Uskutečněné srovnání

Po sumarizaci předchozích možností srovnávání jsem se rozhodl srovnávat filtry či experimenty na několika úrovních. Prvním srovnáním bude ohleduplné srovnání přesností *Acc*, následovat bude vážená přesnost *W Acc* pro poměr 1/50. Pro tento poměr ještě zobrazím procento chybovosti *W Err*, jež je *1 - W Acc*. Jako doplňkový parametr pro porovnávání použiji velikost (1 - ROCA).

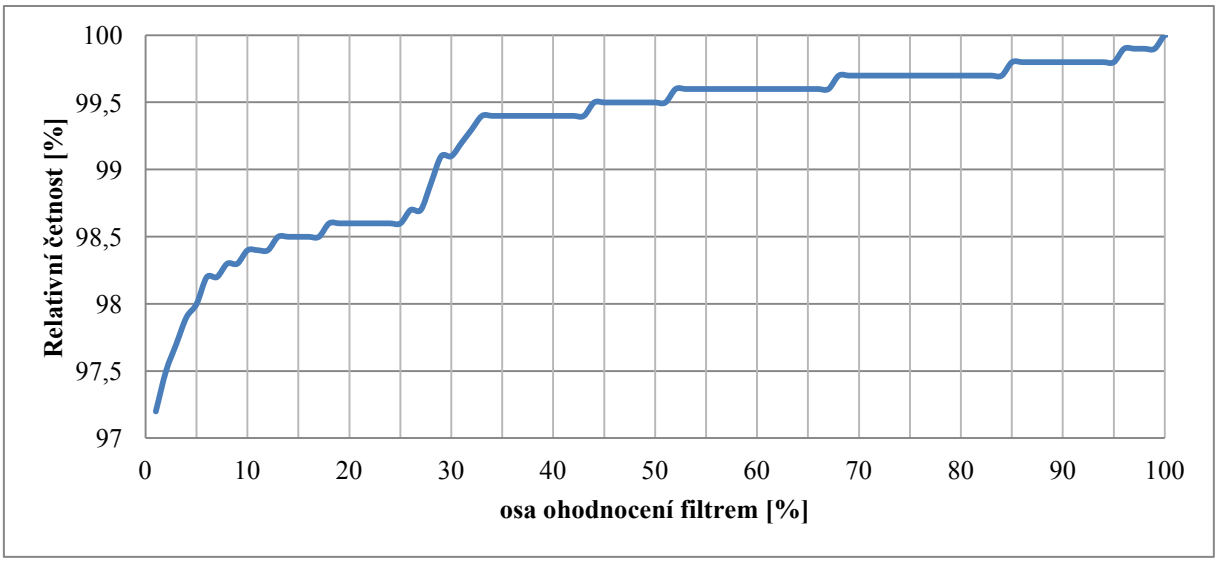
Srovnání filtrů bude probíhat pouze na jednom typu dat a testování experimentování bude rozděleno na 3 typy dat. Pro srovnání bude využit set TREC05 s využitím vyexportovaného textu ze zpráv, pro experimenty bude dále využit ještě set SpamAssasin v podobě vyexportovaného textu i celého emailu na otestování poznatků z testu na předchozí sadě.

5.3. Základní srovnání

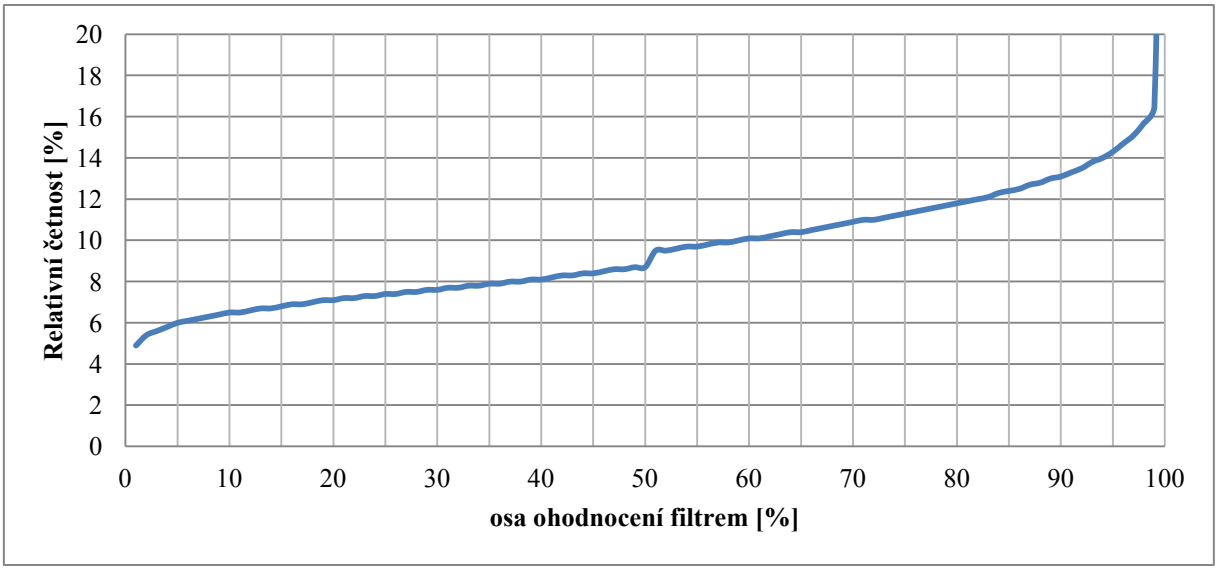
V této podkapitole se budu věnovat srovnání filtrů v jejich základní podobě a základním nastavením. Budou podrobeny křížovému testování na emailové sadě TREC05. Zároveň tyto výsledky budou sloužit pro porovnávání vůči výsledkům experimentů v následující podkapitole.

5.3.1. Rozložení emailů na ose u Bayesovské kombinace

Pokud pohlédneme na *graf č. 2* a *graf č. 3*, vidíme rozložení reálných hamů a spamů na ose po ohodnocení filtrem po otestování celého křížového testu. Tento filtr má svůj práh na hodnotě 0,9 (90%). Při zkoumání grafů vidíme, že hamy jsou v tomto případě velmi jednoduše identifikovány a do hodnoty 0,1 se nachází již přes 98% emailů, což můžeme přičítat stylu výpočtu hodnoty, jelikož počítá pravděpodobnost na spam a na hamy nebere ohled.



Graf č. 2: Relativní kumulativní četnost hamů na klasifikační ose Bayesovské kombinace při testu na sadě TREC05

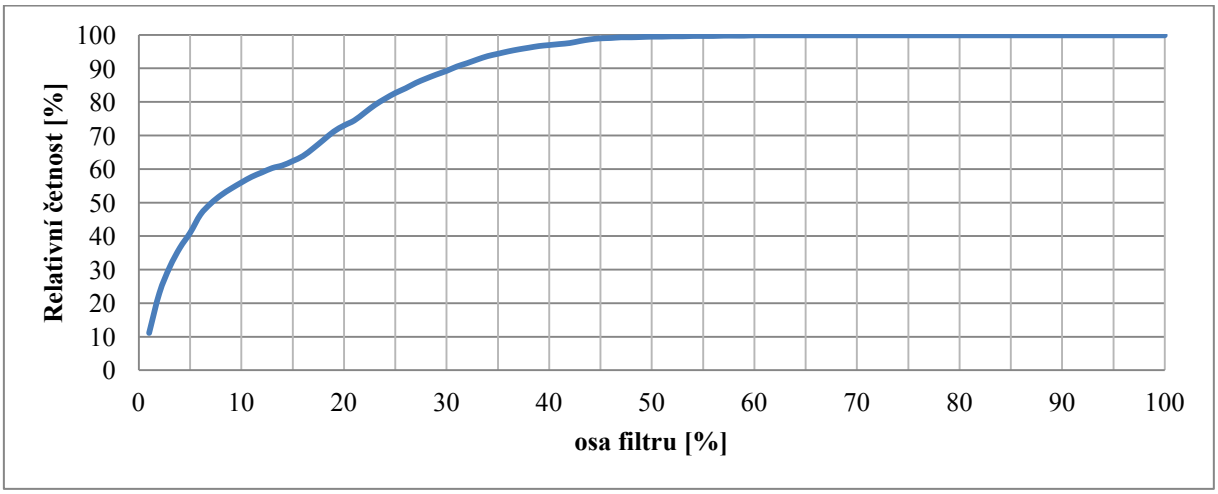


Graf č. 3: Relativní kumulativní četnost spamů na klasifikační ose Bayesovské kombinace při testu na sadě TREC05

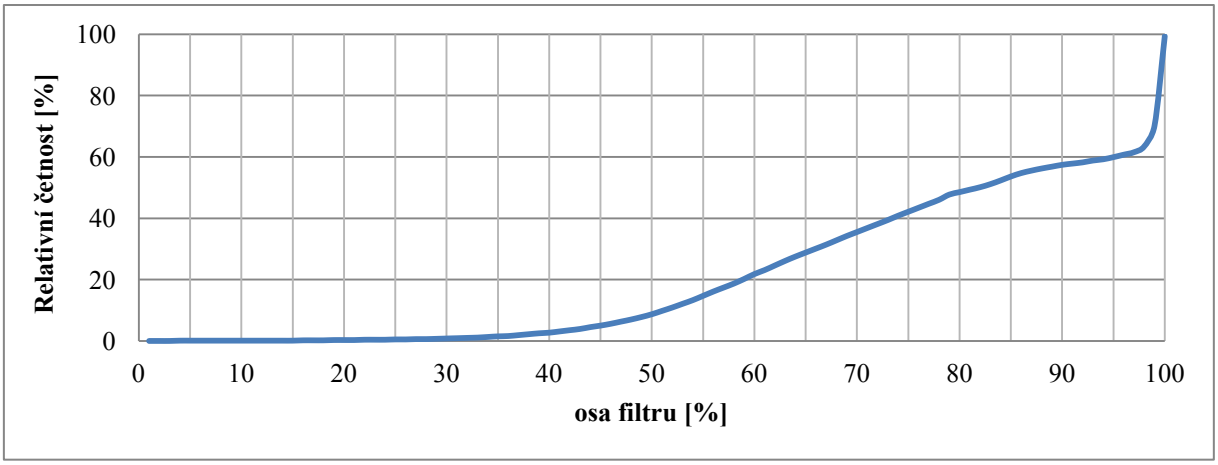
Spamy jsou však rozloženy více na ose a celých 5% z nich je detekováno jako ham se 100% pravděpodobností, dále by se dalo říci, že od hodnot 0,05 do 0,9 narůstá jeho výskyt konstantně. To a také poměr hodnot ve spamu a hamu nabízí možnost uživateli nastavit jiný práh a za malou cenu ztráty hamu vyfiltrovat větší objem spamu.

5.3.2. Rozložení emailů na ose u Robinsonového geometrického průměru

Robinsonův geometrický průměr se oproti Bayesovské kombinaci při výpočtu spoléhá na výpočet hodnot také na pravděpodobnosti hamu, což můžeme vidět již na způsobu výpočtu hodnot a také na rozložení emailů na ose *viz. grafy č. 4 a 5*. Jak můžeme vidět, filtr emaily rozprostírá poměrně neskokově, u hamu kolem hodnoty 0,5 jsme velmi blízko 100% hamů, v opačném směru u hodnoty 0,4 spamy svou četnost dotahují pomalu k nule.



Graf č. 4: Relativní kumulativní četnost hamů na ose Robinsonového geometrického průměru při testu na sadě TREC05

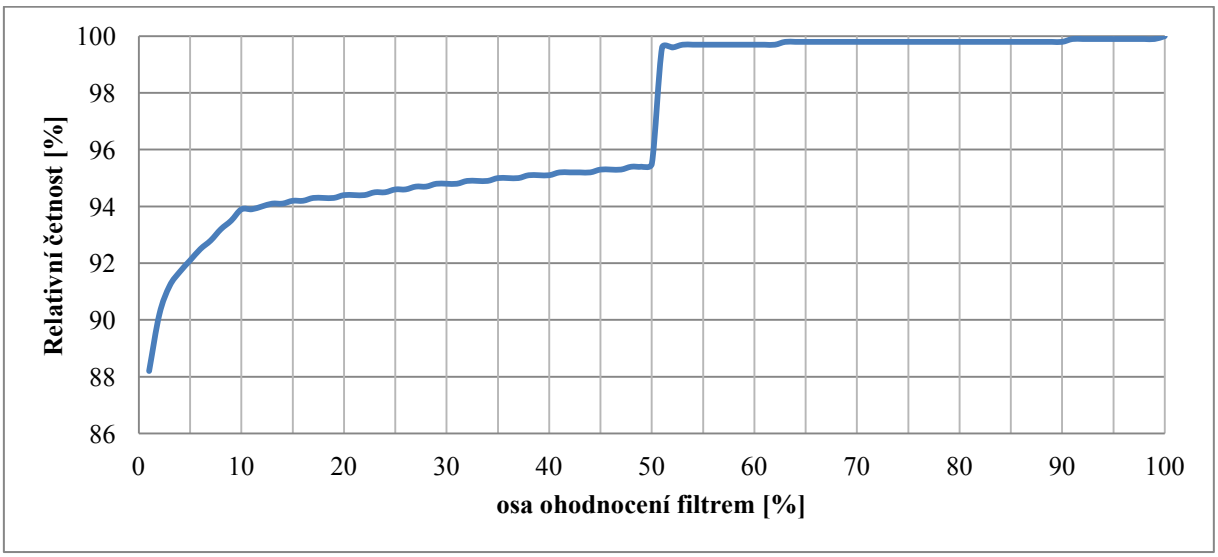


Graf č. 5 Relativní kumulativní četnost spamů na ose Robinsonového geometrického průměru při testu na sadě TREC05

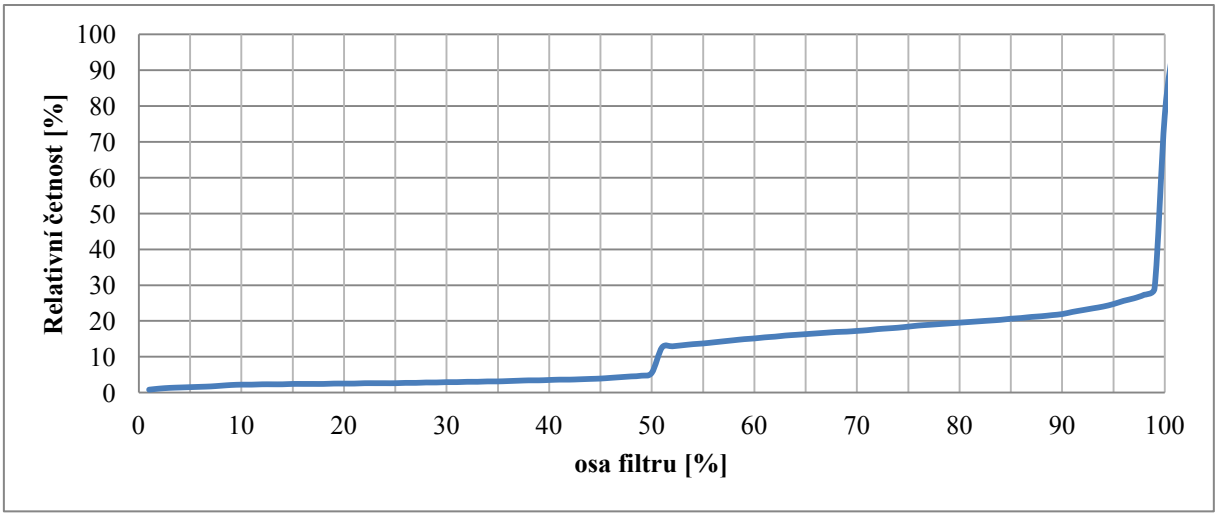
Zde lze opět vidět, že pokud by uživatel pohyboval s prahem detekce, který je optimálně určen na 0,55, a bude chtít obětovat více hamů, může tak odstranit větší množství spamů za menší cenu špatně detekovaných hamů.

5.3.3. Rozložení emailů na ose u Fisher-Robinsonového inversního chí-kvadrátu

Oproti Bayesovské kombinaci se také Robinsonův chí-kvadrát při výpočtu spoléhá na výpočet pravděpodobnosti spamu tak i hamu, což lze opět vidět na způsobu výpočtu, ale také na rozložení emailů na ose. Díky tomu je jeho práh, kdy označí email za spam již 0,55. S pohledem na g*raf č. 6* a g*raf č. 7* vidíme, že rozpoložení na ose je úplně jiné než v předchozích případech.



Graf č. 6: Relativní kumulativní četnost hamů na ose Robinsonového geometrického průměru při testu na sadě TREC05



Graf č. 7: Relativní kumulativní četnost spamů na ose Robinsonového geometrického průměru při testu na sadě TREC05

Velmi výrazným faktorem je příbytek v obou grafech, jak spamů tak hamů, v okolí hodnoty 0,5 a jemně výše. Dalo by se to definovat jako neutrální pole, kde filtr neumí rozhodnout, zda je o spam nebo ham. Práh zde vypadá dobře zvolen, jelikož kdybychom ho posunuli lehce k nule, detekce spamu by sice dosáhla téměř 10% zlepšení, nicméně za cenu 5% hamů, což je moc velká cena.

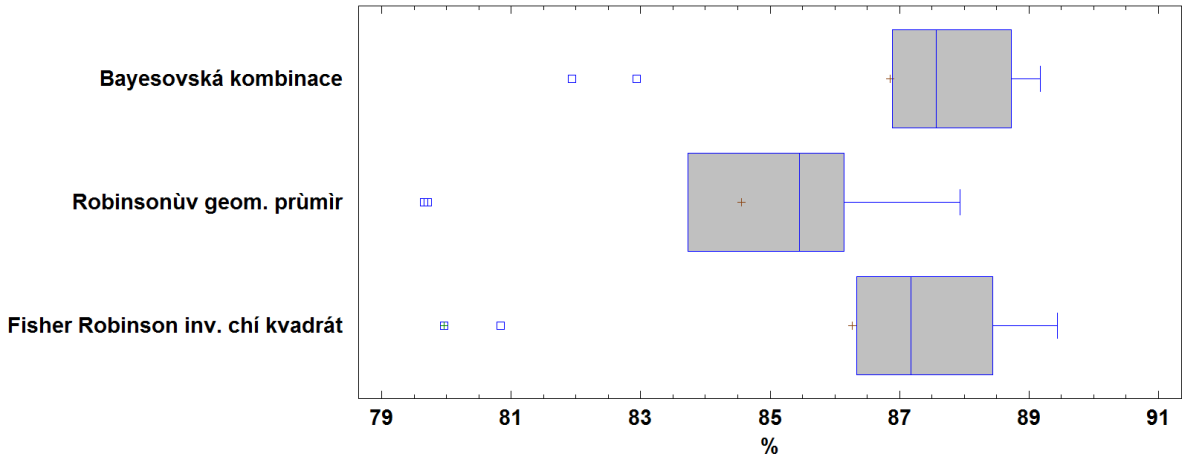
5.3.4. Srovnání výsledků u křížového testu nad setem TREC05

Po provedení křížového testu nad sadou TREC05 jsem získal výstupní data, jaké můžeme vidět v *tabulce č. 10*. Všechny výsledky jsou zprůměrovány z výsledku 10 složek.

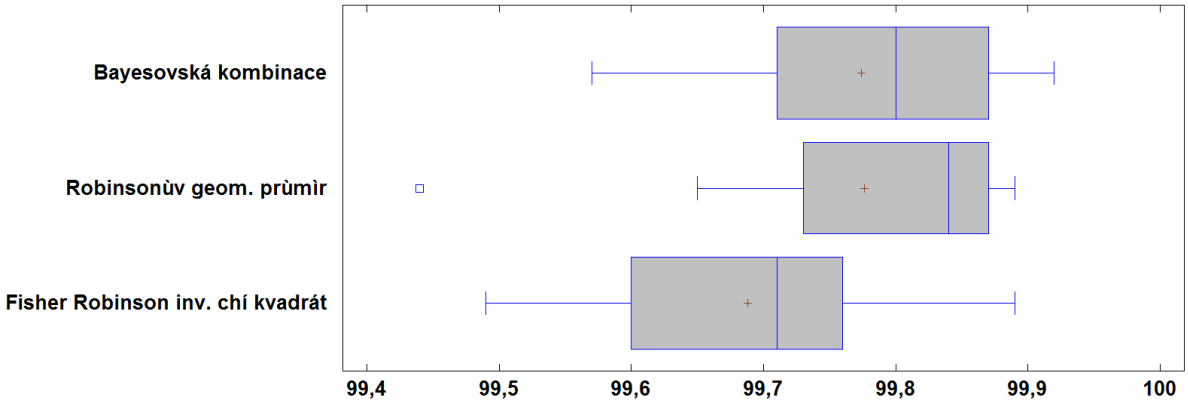
Filtr	Acc	W Acc 1/50	W Err 1/50	1-ROCA
Bayesovská kombinace	92,45	0,9944	0,0056	2,611
Robinsonův geom. Průměr	91,15	0,9939	0,0061	0,503
Robinson-Fisher	92,08	0,9935	0,0065	0,953

Tabulka č. 10: Výsledky filtrů po křížovém testu v základním stavu při testu na sadě TREC05

Můžeme vidět, že z pohledu klasické přesnosti filtry v základním nastavení dosahují všechny kolem 92% úspěšnosti. Nejlepším výsledkem se pyšní Bayesovská kombinace, dále Robinson Fisher inversní chí-kvadrát a jako poslední Robinsonův geometrický průměr. To proč tomu tak je můžeme vydedukovat z krabicových g*rafů č. 8 a 9*. Jelikož je přesnost počítána celkově a hamy jsou téměř všechny detekovány oproti spamům, výsledek ovlivňuje výsledek spamu. Jak můžeme vidět, průměrné hodnoty jsou seřazeny stejně, jako je náš výsledek.



Graf č. 8: Krabicový graf úspěšnosti detekce spamu všech 3 filtrů v procentech při testu na sadě TREC05



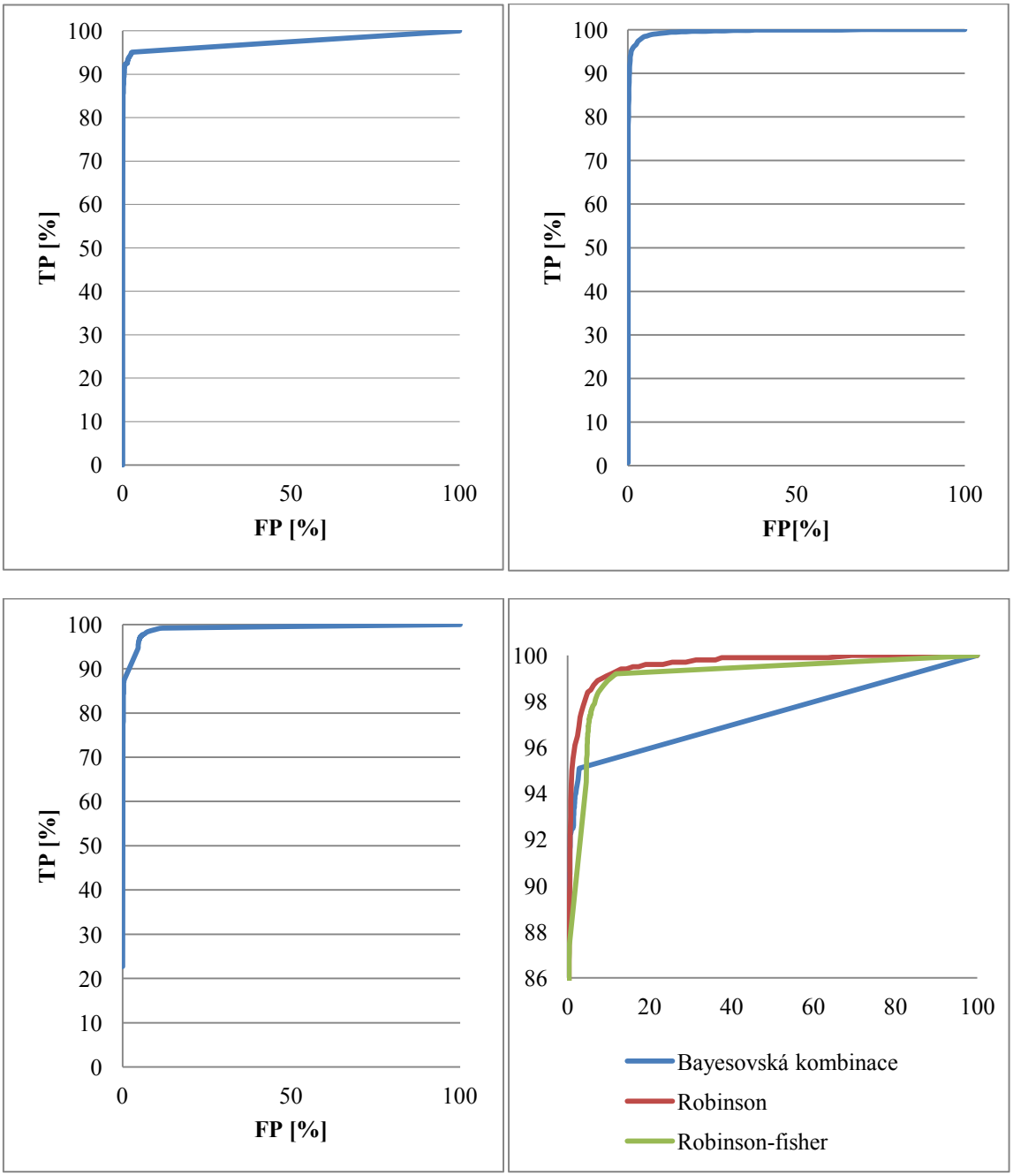
Graf č. 9: Krabicový graf úspěšnosti detekce hamu všech 3 filtrů v procentech při testu na sadě TREC05

Nicméně tato data nám přímo nepovídají o poměru chyb hamů a spamů. Při vyhodnocení vážené přesnosti vidíme, že jsou si filtry vyrovnané a již jinak seřazeny. Důvodem je přidaná váha výsledkům hamů. Pokud opět pohlédneme na *graf č. 8 a č. 8*, vidíme, že u Robinsonova geometrického průměru u úspěšnosti detekce hamů máme nejlepší průměr, který je ještě velmi ovlivněn odlehlým pozorováním, soudě podle vzdálenosti průměru od mediánu. I když máme přidanou váhu k hamům, stále to nestačilo na Bayesovskou kombinaci, která má téměř stejný průměr úspěšnosti hamu, avšak ve spamu dosáhla mnohem lepší detekce.

Jako posledním porovnávacím faktorem je 1-ROCA. V *Tabulce č. 10* vidíme, že nejmenšího a tedy nejlepšího výsledku dosahuje Robinsonův geometrický průměr s dvojnásobně lepším výsledkem než druhý Chí-kvadrát a 5x lepším než Bayesovská kombinace.

Jestliže pohlédneme na *graf č. 10*, vidíme zde ROC křivky všech filtrů na celkovém poli. Již zde je vidět, že filtr s nejlepším ohodnocením má jemně zaoblenou křivku táhnoucí se velmi blízko hornímu levému bodu plochy, kde je ideální stav. Při pohledu na poslední z grafů dostáváme srovnání daných křivek v detailu a je zde viditelné, kde se tvoří rozdíly v AUC, tedy plochou pod křivkou.

Co se týče křivky pro Bayesovskou kombinaci, její tvar a zalomení je pravděpodobně tvořeno její vlastností výpočtu pouze spamové hodnoty. Je to však moje teze, kterou nemohu potvrdit.



Graf č. 10: Čtveřice grafů na porovnání křivek ROC při testu na sadě TREC05 v pořadí zleva doprava posléze dolů: Bayesovská kombinace, Robinsonův geom. průměr, Robinson Fisher inv. chí-kvadrát. Jako poslední je záběr na detail důležité části všech křivek v jednom grafu

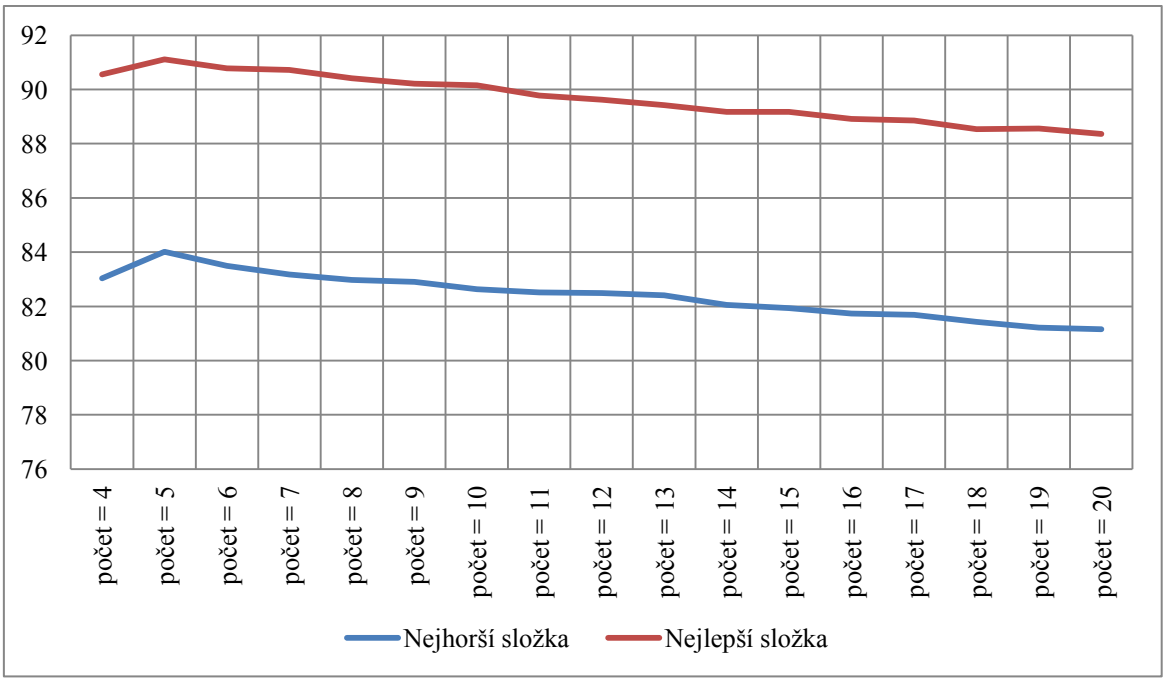
5.4. Vlastní experimenty na velké sadě

Nápady na experimentování na filtrech jsem bral z vlastních úvah, či z nápadů o které se už někdo pokoušel [3]. Úvahy jsou implementovány a je zkoumáno, zda zlepšují či naopak zhoršují detekci spamu.

5.4.1. Experiment testování počtu významných slov

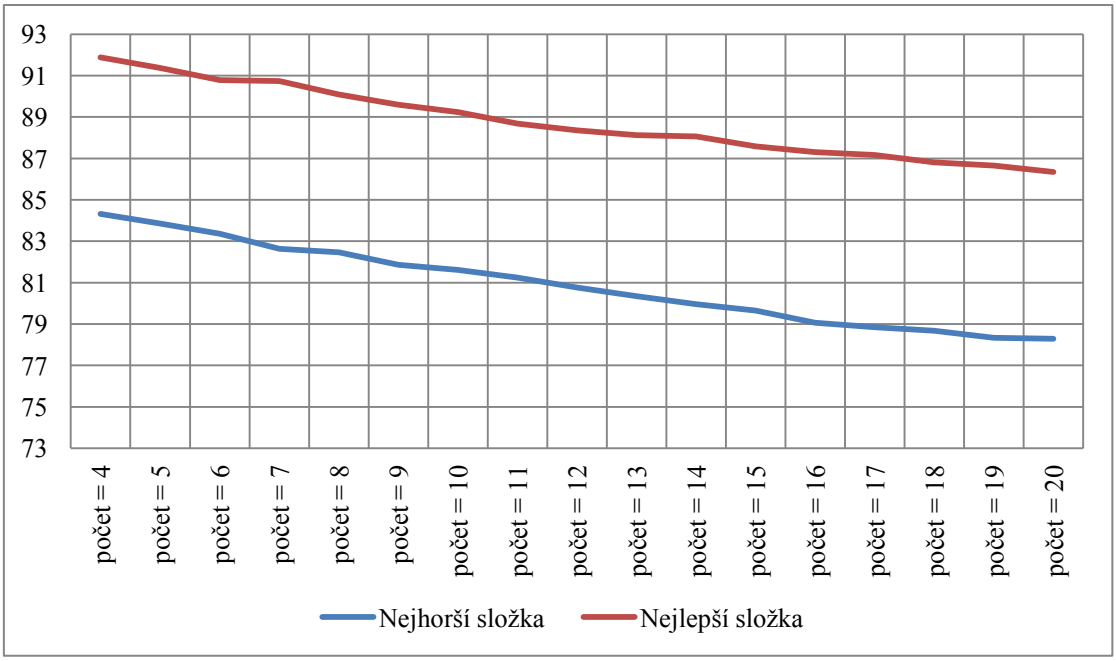
Tento experiment by měl zjistit, zda není výhodnější využívat méně či více významných slov. Proto jsem otestoval, jak vypadají výsledky pro první 2 filtry, které používají počet prvků 15, na velikosti okolo něj.

Aby šlo vidět, jak se mění úspěšnost detekce spamu, vybral jsem složku s nejhoršími a nejlepšími výsledky a změny při různých počtech významných slov zaznamenal do *grafu č. 11* pro Bayesovskou kombinaci a do *grafu č. 12* pro Robinsonův geometrický průměr.



Graf č. 11: Změna procentuální úspěšnosti detekce spamu při změně počtu významných slov na nejhorší a nejlepší složce u Bayesovské kombinace při testu na sadě TREC05

Jak je vidět na prvním ze jmenovaných grafů, v počtu 5 dosahuje výsledek maxima a pak již pomalu stagnuje. Na druhém grafu je maximum hned v počtu 4 a dále křivka pouze klesá. Pokud bychom se měli řídit pouze tímto, bylo by lepší využít méně slov než více. Při testu úspěšnost hamů byla kolísavá a úspěšnost se lišila pouze v desetinách procenta, nicméně i toto může znamenat hodně.



Graf č. 12: Změna procentuální úspěšnosti detekce spamu při změně počtu významných slov na nejhorší a nejlepší složce u Robinsonového geometrického průměru při testu na sadě TREC05

Pro nejlepší vyhodnocení nám však bude sloužit t*abulka č. 11 a 12*, kde jsem vybral pro mne nejzajímavější body 5, 10 a 15 pro oba testované filtry.

Počet slov	Acc	W Acc 1/50	1-ROCA
5	0,17	-0,0001	0,914
10	0,06	-0,0001	0,39955
15	92,75	0,9946	2,6106

Tabulka č. 11: Změna charakteristik při testu počtu významných slov na 5 a 10 v porovnání s klasickými 15 slovy u Bayesovské kombinace na sadě TREC05.

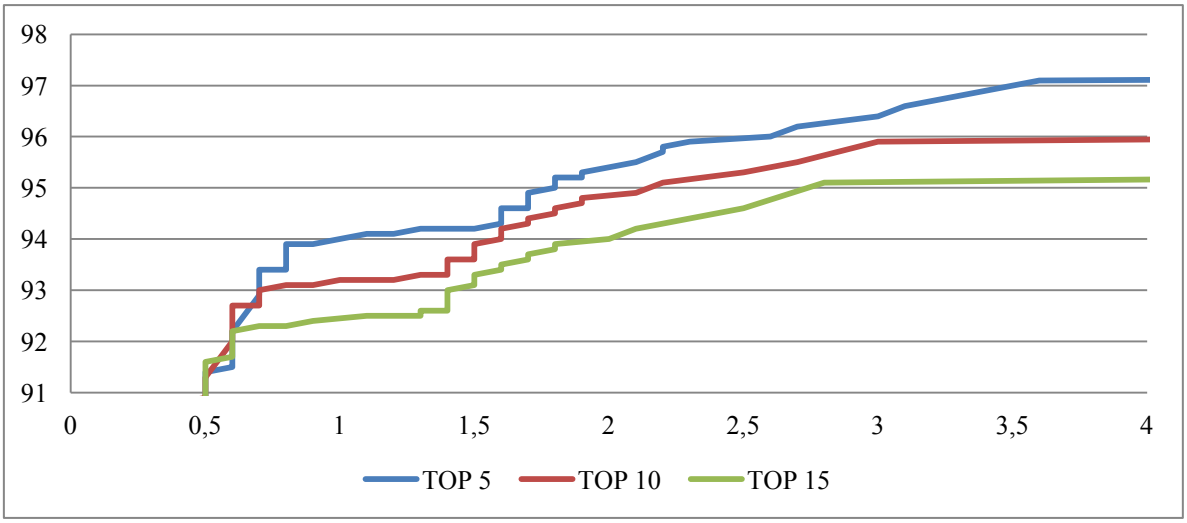
Jak vidíme, přesnost nám s menší počtem slov lehce stoupá, kdežto vážená přesnost klesla minimálně. To nám vypovídá o tom, že se lépe detekovaly spamy, kdežto hamy se v daném poměru lehce zhoršily. Nejzajímavějším ukazatelem je 1-ROCA, která u 5 slov dosáhla velmi velkého zlepšení. To proč tomu tak je vidíme na g*rafu č. 13*, kde lze vidět, že téměř na celém grafu má charakteristika pro počet 5 navrch.

Počet slov	Acc	W Acc 1/50	1-ROCA
5	0,97	0,0004	-0,0732
10	0,13	0,0000	0,0162
15	89,31	0,9932	0,5034

Tabulka č. 12: Změna charakteristik při testu počtu významných slov na 5 a 10 v porovnání s klasickými 15 slovy u Robinsonového geometrického průměru na sadě TREC05.

Stejně jako při Bayesovské kombinaci se Robinsonův geometrický průměr choval podobně viz *tabulka č. 12*. Vidíme, že v přesnosti jsme se snižováním dosáhli lehce lepších výsledků, nicméně vážená přesnost se téměř neměnila. 1-ROCA dosáhla nejlepšího hodnocení v počtu 10 slov, avšak pouze minimálního.

Z tohoto pohledu by se dalo říci, že zde až tak nezáleží na počtu slov. Tato úprava by se měla užít maximálně při individuální potřebě, která by uplatnila snížení nebo zvětšení počtu významných slov, v normálním použití nemá dle mne významného uplatnění.



Graf č. 13: ROC křivka pro 3 varianty počtů významných slov u Bayesovské kombinace v nejzajímavější části plochy při testu na sadě TREC05.

Pro třetí filtr nazván Robinson Fisher inversní chí-kvadrát jsem musel udělat experiment trochu odlišně a to posouvat hranice od krajů osy. Ve svém experimentu jsem otestoval rozdíly pro hodnoty 0,015, 0,01 a 0,005.

Rozsah od kraje	Acc	W Acc 1/50	1-ROCA
0,005	-0,04	-0,0008	-0,1244
0,01	92,08	0,9935	0,9533
0,015	-0,33	0,0005	0,0327

Tabulka č. 13: Změna charakteristik při testu rozmezí významných slov na 0,005 a 0,01 v porovnání s klasickou hodnotou 0,015 u Robinson Fisher inversního chí-kvadrátu na sadě TREC05.

V *tabulce č 13* vidíme, že rozsah zde již více ovlivňuje jak váženou přesnost, tak hodnotu 1-ROCA. Dle těchto výsledků to vypadá, že větší rozsah lehce sníží detekci spamu avšak v poměru lépe vyhodnotí hamy. Rozsah bych doporučil u tohoto filtru změnit.

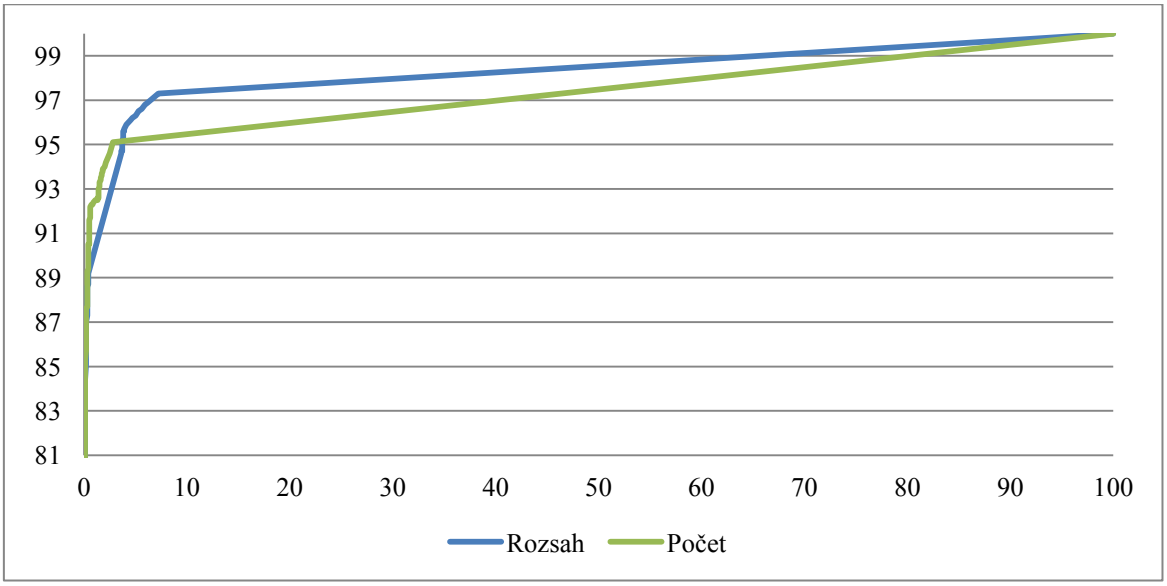
Jako další experiment této časti jsem se rozhodl prozkoumat, jak si povedou první 2 filtry s definovaným rozmezím pro významná slova místo počtu významných slov.

Typ	Acc	W Acc 1/50	1-ROCA
počet	92,45	0,9944	2,6106
rozsah	0,45	-0,0004	0,89415

Tabulka č. 14: Změna charakteristik při testu záměny počtu významných slov s rozmezím na Bayesovské kombinaci na sadě TREC05.

Tabulka č. 15 nám ukazuje, že u Bayesovské kombinace došlo ke zlepšení přesnosti, nicméně se nám zhoršila vážená přesnost, což je znakem lepší detekce spamu avšak zhoršení detekce hamu. Nicméně velmi zajímavý číslem je rozdíl u hodnoty 1-ROCA, kde došlo k velkému zlepšení. Proč

tomu tak je můžeme vidět na *grafu č. 14*, kde lze vidět, že za touto změnou stojí trochu jiné rozložení křivky.

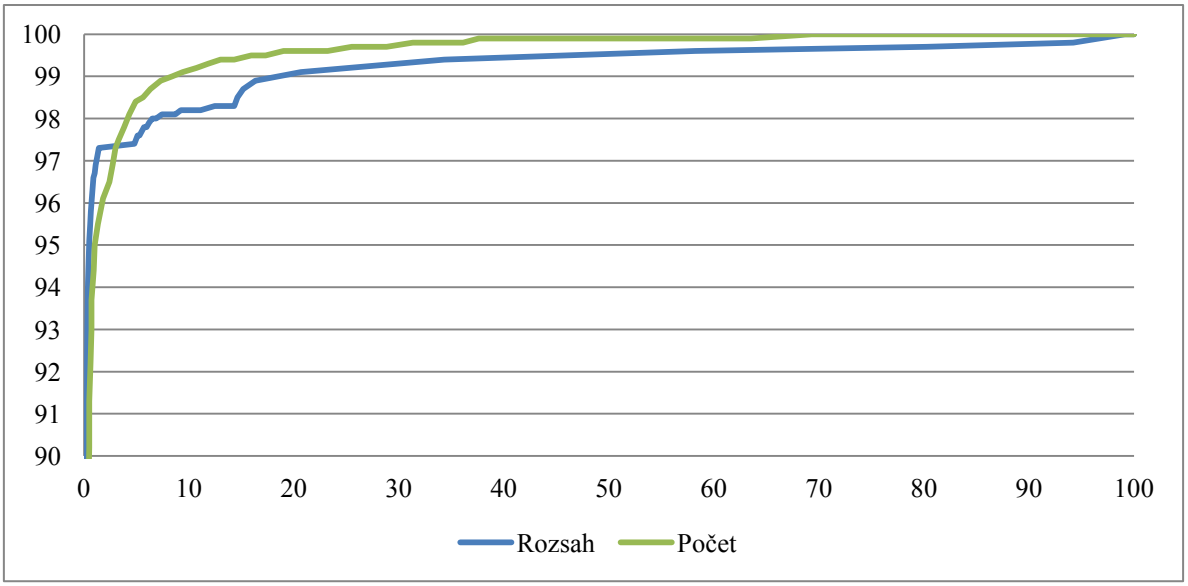


Graf č. 14: ROC křivky pro Bayesovskou kombinaci a varianty počtu i rozsahu významných slov při testu na sadě TREC05.

Typ	Acc	W Acc 1/50	1-ROCA
počet	91,15	0,9939	0,50335
rozsah	1,00	0,0001	-0,30115

Tabulka č. 15: Změna charakteristik při testu záměny počtu významných slov s rozmezím na Robinsonovém geometrickém průměru na sadě TREC05.

Tabulka č. 15 nám ukazuje, že na Robinsonův geometrický průměr měl experiment opačný jev jako na Bayesovskou kombinaci. Přesnost se sice zlepšila, ale vážená přesnost dosáhla také zlepšení, takže se zlepšila detekce spamu a také hamu v poměru příznivějším. Ovšem 1-ROCA získala horší výsledek. Na *grafu č. 15* vidíme, že se celkově změnila charakteristika filtru. Změny nastávají hlavně ze začátku křivky, kdy je lepší detekce pro rozsah, později však lepších statistik dosáhne varianta s počtem.



Graf č. 15: ROC křivky pro Robinsonův geometrický průměr a varianty počtu i rozsahu významných slov.

5.4.2. Experiment s čísly jako oddělovači

Tento experiment má za úkol zkusit, zda čísla nesou nějakou důležitou informaci pro nás či nikoliv. Pokud se nad tím hlouběji zamyslíme, teoreticky by čísla z hlediska samostatné informace neměly pro spamový filtr mít význam. Proto jsem udělal tento experiment, zda tomu opravdu tak je a vypouštím je ze slov a vytvořím z nich oddělovače.

Tabulka č. 16 nám ukazuje, že přesnost, která je pro nás nejméně důležitá, stoupla pouze u druhého filtru, a vážená přesnost i 1 - ROCA dosáhly všude horších výsledků. Rozdíly jsou však poměrně malé, takže se můžeme domnívat, že pouze některá čísla sebou můžou nést důležitou informaci, avšak ta informace tam pravděpodobně je. Ve výsledku také ale musíme brát v potaz to, že díky číslům se zvětší databáze slov se statistikami. Rozumnější volbou tedy pravděpodobně bude čísla opravdu použít jako oddělovače, jelikož výsledek není tolik velký v poměru k zvětšení datasetu slov.

Filtry a změna statistik	Acc	W Acc 1/50	1-ROCA
Bayesovská kombinace	-0,12	-0,0003	-0,118
Robinsonův geom. Průměr	0,01	-0,0001	-0,060
Robinson-Fisher	-0,55	-0,0003	-0,111

Tabulka č. 16: Změna výsledků statistiky po aplikování experimentu s čísly při testu na sadě TREC05. Záporné hodnoty značí zhoršení, kladné zlepšení.

5.4.3. Experiment s velkými a malými písmeny

Další z experimentů má za úkol prozkoumat, jestli sensitivita na písmena dokáže změnit výsledky filtrů. Při této úpravě dojde k zvětšení velikosti datasetu, jelikož vznikne spousta variací slov, která si bude udržovat vlastní statistiky výskytů.

V tabulce č. 17 vidíme, že ve všech statistikách došlo k velkým zlepšením. Z pohledu přesnosti došlo ve všech filtrech ke zlepšení. U prvních dvou o cca 1,5%, Robinson-Fisher se zlepšil pouze o

0,7%. Nás však více zajímá vážená přesnost, kde všechny filtry zlepšily svou hodnotu téměř o stejný díl. 1 - ROCA také dosáhla zlepšení, nejrazantnějšího u Bayesovské kombinace.

Ve světle těchto výsledků, i když vzniká dosti variací slov, myslím, že toto je úspěšný experiment, který je vhodný pro reálné využití.

Filtry a změna statistik	Acc	W Acc 1/50	1-ROCA
Bayesovská kombinace	1,66	0,0007	0,336
Robinsonův geom. Průměr	1,71	0,0008	0,128
Robinson-Fisher	0,77	0,0006	0,157

Tabulka č. 17: Změna výsledků statistiky po aplikování experimentu s velkými a malými písmeny při testu na sadě TREC05. Záporné hodnoty značí zhoršení, kladné zlepšení.

5.4.4. Experiment s vykřičníky a otazníky

Tento experiment je trochu specifický, jelikož mne napadl při pohledu na mou starou přeplněnou emailovou schránku, v níž převládají spamy a texty použité v nich se často vyznačují upoutáním pozornosti pomocí těchto znaků.

Tabulka č. 18 nám prozrazuje, že ke zlepšení opravdu došlo. Ve všech filtrech u přesnosti dochází ke zlepšení. U Robinson-Fishera ve vážené přesnosti dostáváme stejný výsledek jako původní, i tak můžeme tvrdit, že tato úprava zvládla zlepšit výsledky, a proto tento experiment beru za užitečný ve všech filtrech.

Filtry a změna statistik	Acc	W Acc 1/50	1-ROCA
Bayesovská kombinace	0,29	0,0004	0,165
Robinsonův geom. Průměr	0,51	0,0003	0,042
Robinson-Fisher	0,27	0,0000	0,084

Tabulka č. 18: Změna výsledků statistiky po aplikování experimentu s vykřičníky a otazníky při testu na sadě TREC05. Záporné hodnoty značí zhoršení, kladné zlepšení.

5.4.5. Experiment s url a emailovými adresami

Jako další experiment je vyhledávání url a emailových adres v textu a jejich speciální zařazení do datasetu. Tento experiment však nemusí mít racionální výsledky, jelikož je testován nad veřejným setem emailů, který má přepsány emailové adresy kvůli ochraně soukromí. Není mi bohužel známo, jakým způsobem byly data nahrazena, zda zde existovala nějaká pravidla, či byly náhrady generovány náhodně.

Filtry a změna statistik	Acc	W Acc 1/50	1-ROCA
Bayesovská kombinace	0,73	0,0006	0,323
Robinsonův geom. Průměr	0,90	0,0008	0,079
Robinson-Fisher	0,46	0,0002	0,098

Tabulka č. 19: Změna výsledků statistiky po aplikování experimentu s url a emailovými adresami při testu na sadě TREC05. Záporné hodnoty značí zhoršení, kladné zlepšení.

Tabulka č. 19 nám prozrazuje, že pravděpodobně nahrazování adres nebylo náhodné a tento experiment pomáhá filtrům ve zlepšení. Ve všech měřících metodách bylo dosaženo zlepšení, zajímavé určitě je, že 1-ROCA u Bayesovské kombinace zlepšila velmi svou hodnotu. Pravděpodobně je to dáno tím, že se podařilo na ose více odsadit spamy a hamy od sebe a vytvořily

se tak příznivější poměry správné detekce a chyb. Výsledky nám tedy naznačují, že experiment má kladný účinek na detekci.

5.4.6. Experiment se sousedícími slovy

Jeden ze zajímavějších experimentů je sledování sousedících slov. Tímto experimentem dodáváme jemně do vyhledávání sledování kontextu, v jakém se slova nachází. Experiment sledoval dvojce, kde nezáleželo na pořadí, poté i aby na pořadí záleželo.

Filtry a změna statistik	Acc	W Acc 1/50	1-ROCA
Bayesovská kombinace	2,38	0,0017	0,372
Robinsonův geom. Průměr	2,81	0,0019	0,113
Robinson-Fisher	-6,84	-0,0018	-1,750

Tabulka č. 20: Změna výsledků statistiky po aplikování experimentu se sousedícími slovy, když nezáleží na pořadí při testu na sadě TREC05. Záporné hodnoty značí zhoršení, kladné zlepšení.

Výsledky v *tabulce č. 20* ukazují, že dosahují v porovnání s ostatníma nejzajímavějších hodnot. Co se týče přesnosti, dosáhli jsme zlepšení víc jak o 2% u prvních dvou filtrů, což svědčí o zlepšení spamové detekce. Změna vážené přesnosti také dosáhla změny, která je velmi výrazná, což naznačuje zlepšení také hamové detekce. Výsledky 1-ROCA také potvrzují lepší rozložení emailů na ose. Výjimkou je filtr Robinson Fisher inversní chí-kvadrát, kde došlo k velmi razantním zhoršením. Dle mých úvah by za to měl být zodpovědný buď výběr významných slov, protože je možné, že se takto dostalo více slov a dvojicí do rozhodování o hodnotě emailu, a jak bylo již otestováno v p*odkapitole 5.4.1*, více slov zhoršuje detekci. Druhou možností, proč tomu tak může být, je samotná logika vzorce filtru.

Filtry a změna statistik	Acc	W Acc 1/50	1-ROCA
Bayesovská kombinace	2,37	0,0016	0,422
Robinsonův geom. Průměr	2,87	0,0020	0,118
Robinson-Fisher	-7,03	-0,0015	-1,870

Tabulka č. 21: Změna výsledků statistiky po aplikování experimentu se sousedícími slovy, když záleží na pořadí při testu na sadě TREC05. Záporné hodnoty značí zhoršení, kladné zlepšení.

Pro variaci testu se sousedícími slovy, kde již záleží na pořadí, je *Tabulka č. 21*. Jak můžeme vidět oproti předchozímu experimentu nedošlo k žádným extrémním změnám, pouze Bayesovská kombinace ještě více zlepšila svou hodnotu 1-ROCA.

Tento experiment se pro první 2 filtry velmi vyvedl, nicméně přináší spoustu dat navíc. Pokud bychom se rozhodli využít tento experiment, musíme počítat s objemnějším historickým datasetem, avšak i s velkým zlepšením filtru. Při vybírání variace doporučuji využít tu, kde nezáleží na pořadí, jelikož nezabere až tolik místa a výsledky jsou téměř totožné. Pro třetí filtr tento experiment nedoporučuji.

5.4.7. Experiment s megaspamem

Megaspam je mým názvem metody, která přidává spamovým slovům, které se objeví v emailu několikrát, váhu a snaží se tedy spamy lépe detekovat. Pro experiment jsem otestoval dvě varianty.

První varianta, jež má výsledky v t*abulce č. 22*, je pro nastavení 3/2, tedy pokud se spam objeví v emailu alespoň 3x, jeho četnost ve významných slovech bude zdvojnásobena. Jak vidíme,

experiment v prvních 2 filtrech dosáhl minimálního zlepšení, Robinson Fisher dosáhl dokonce zhoršení.

Filtry a změna statistik	Acc	W Acc 1/50	1-ROCA
Bayesovská kombinace	0,14	0,0000	0,053
Robinsonův geom. Průměr	0,34	0,0001	0,006
Robinson-Fisher	-0,09	-0,0004	-0,001

Tabulka č. 22: Změna výsledků statistiky po aplikování experimentu s megaspamem v nastavení 3/2 při testu na sadě TREC05. Záporné hodnoty značí zhoršení, kladné zlepšení.

Filtry a změna statistik	Acc	W Acc 1/50	1-ROCA
Bayesovská kombinace	0,31	-0,0005	0,136
Robinsonův geom. Průměr	0,68	-0,0002	-0,003
Robinson-Fisher	-0,46	-0,0009	-0,027

Tabulka č. 23: Změna výsledků statistiky po aplikování experimentu s megaspamem v nastavení 2/3 při testu na sadě TREC05. Záporné hodnoty značí zhoršení, kladné zlepšení.

Druhá varianta má za úkol vyzkoušet, zda nastavení 2/3 dokáže zlepšit detekci. V *tabulce č. 23* můžeme vidět, že opět poslední filtr test zhoršuje. U prvních dvou vidíme, že přesnost se zlepšila, nicméně vážená se zhoršila. To ukazuje, že detekce spamu se zlepšila avšak na úkor hamu. 1-ROCA zlepšila svou hodnotu pouze u Bayesovské kombinace, avšak ne moc výrazně.

Implementace tohoto experimentu nijak nepřitěžuje velikosti datasetu a pouze nevýrazně při zpracování, nicméně výsledky ani jedné varianty nejsou moc ohromující, a proto soudím, že je zbytečné tuto funkci implementovat.

5.4.8. Experiment kombinací úspěšných experimentů

Nastavení	Bayes č.1	Bayes č.2	Robinson č.1	Robinson č.2	R-Fisher č.1	R-Fisher č.2
Počet nebo rozsah	počet	počet	počet	počet	rozsah	rozsah
Hodnota	15	5	15	10	0,01	0,015
Číslo odděluje	NE	NE	NE	NE	NE	NE
Velká malá písmena	ANO	ANO	ANO	ANO	ANO	ANO
Vykřičníky otazníky	ANO	ANO	ANO	ANO	ANO	ANO
URL a email	ANO	ANO	ANO	ANO	ANO	ANO
Dvojce slov	ANO	ANO	ANO	ANO	NE	NE
Záleží na pořadí	NE	NE	NE	NE	NE	NE
Megaspam	NE	NE	NE	NE	NE	NE

Tabulka č. 24: Tabulka nastavení filtrů pro kompletní experiment pouze se všemi účinnými částmi.

Jakožto jeden z posledních testů je spojení úspěšných experimentů v jeden. Při tomto experimentu bude zajímavé, jak moc se zlepší detekce či 1-ROC, jelikož některé experimenty můžou na sebe působit negativně, či ovlivňovat pouze stejné emaily.

Pro filtry použiji nastavení definované v *tabulce č. 24*. Pro každý filtr jsem vytvořil 2 varianty. Každá varianta bude otestována na 2 sadách velkého setu, abychom viděli, zda nejde o shodu náhod.

Filtry a změna statistik	Acc	W Acc 1/50	1-ROCA
Bayesovská kombinace č.1	3,24	0,0019	0,524
Robinsonův geom. Průměr č. 1	3,42	0,0020	0,093
Robinson-Fisher č. 1	1,07	0,0005	0,212

Tabulka č. 25: Změna výsledků po aplikování experimentu pro první variantu každého filtru z *tabulky č.24* při testu na sadě TREC05

První sada testů proběhla v jejich klasickém nastavení významných slov. Výsledky můžeme vidět v *tabulce č. 25*, kde vidíme, že se podařilo vylepšit vše. Přesnosti se u prvních dvou filtrů zvedly o více jak 3%, a vážené přesnosti o 0,002, což je velmi dobrý výsledek. Robinson-Fisher však dosáhnul pouze menšího zlepšení, což může být dáno také tím, že u něj většina experimentů nedopadala nejlépe a hůře než u zbylých 2. Hodnota 1-ROCA byla vylepšena ve všech případech, nejvíce u Bayesovské kombinace.

Filtry a změna statistik	Acc	W Acc 1/50	1-ROCA
Bayesovská kombinace č. 2	2,15	0,0012	-0,174
Robinsonův geom. Průměr č. 2	3,40	0,0021	0,000
Robinson-Fisher č. 2	0,76	0,0008	0,189

Tabulka č. 26: Změna výsledků po aplikování experimentu pro druhou variantu každého filtru z *tabulky č.24* při testu na sadě TREC05

Druha sada testů proběhla v zajímavých místech počtu nebo rozsahu významných slov. Výsledky jsou situovány v *tabulce č. 26* a můžeme je porovnávat ihned s výsledky první sady v *tabulce č. 25*. Ačkoliv si Bayesovská kombinace s 5 významnými slovy na klasické verzi vedla stejně jako s 15 slovy, avšak 1-ROCA se velmi vylepšila *viz. tabulka č. 11*, nyní došlo s více experimenty k obratu a dosáhli jsme zlepšení poměrně velkého. Velkou změnou je však 1-ROCA kde jsme se z +0,8 posunuli na -0,2. V porovnání s testem z první sady vidíme, že výrazně lepších výsledků dosahovala varianta č. 1. a to ve všech metrikách. U Robinsonového průměru můžeme pozorovat, že rozdíl ve významných slovech ani zde nehraje téměř žádnou roli, jelikož výsledky jsou si velmi podobné. Robinson-Fisher v testu klasického nastavení s experimentem na rozsah slov vyšel lépe v této variantě *viz. tabulka č. 13*, pokud však porovnáme výsledky těchto testů zjistíme, že se efektivity skoro srovnaly.

Jakožto hromadné zhodnocení můžeme brát *tabulku č.27*, kde jsou k nahlédnutí celkové výsledky filtrů. Pokud bychom chtěli hodnotit podle přesnosti, zjistíme že nyní by nejlepším filtrem byla Bayesovská kombinace č.1, dále také druhá varianta a obě varianty Robinsonového geometrického průměru. Bayesovská kombinace č.1 také ovládla váženou přesnost, která je pro nás důležitější a vidíme, že snížila chybovost od klasické verze o třetinu. Podobného zlepšení dosáhly i experimenty Robinsonového průměru. Experimenty Robinson-Fisher inversního chí-kvadrátů dosáhly zlepšení, avšak jen symbolického oproti ostatním filtrům.

Filtr	Acc	W Acc 1/50	W Err 1/50	1-ROCA
Bayesovská kombinace	92,45	0,9944	0,0056	2,611
Bayesovská kombinace č. 1	95,69	0,9963	0,0037	2,08735
Bayesovská kombinace č. 2	94,61	0,9956	0,0044	2,78495
Robinsonův geom. Průměr	91,15	0,9939	0,0061	0,503
Robinsonův geom. Průměr č. 1	94,57	0,9959	0,0041	0,40975
Robinsonův geom. Průměr č. 2	94,56	0,9959	0,0041	0,50255
Robinson-Fisher	92,08	0,9935	0,0065	0,953
Robinson-Fisher č. 1	93,15	0,9940	0,0060	0,74125
Robinson-Fisher č. 2	92,84	0,9943	0,0057	0,76355

Tabulka č. 27: Porovnání filtrů a kombinací experimentů při testu na sadě TREC05

Jako další experiment této části je prozkoumat, zda budou výsledky podobné se stejnými daty, pouze jinak rozdělenými do složek. Výsledek tohoto testu můžeme vidět v *tabulce č. 28*, kde je základ i experimenty proveden znova. Nyní můžeme vidět, že soubory v těchto složkách byly pro testování příhodněji rozděleny a ve všech případech dosahují lepších výsledků. Co se týče prováděného experimentu s vylepšujícími experimenty, výsledky dopadly jako v předchozím testu podobně a nedošlo k téměř žádnému zvratu. Jediné, co se výrazně liší je hodnota 1-ROCA u Bayesovské kombinace č. 2, kde došlo k výraznému zlepšení a hodnota oproti předchozímu testu předčila základní nastavení.

Filtr	Acc	W Acc 1/50	W Err 1/50	1-ROCA
Bayesovská kombinace	93,35	0,9953	0,0047	2,2842
Bayesovská kombinace č. 1	96,28	0,9966	0,0034	1,7295
Bayesovská kombinace č. 2	95,48	0,9962	0,0038	2,18165
Robinsonův geom. Průměr	92,26	0,9944	0,0056	0,42135
Robinsonův geom. Průměr č. 1	95,49	0,9964	0,0036	0,3187
Robinsonův geom. Průměr č. 2	95,46	0,9964	0,0036	0,3987
Robinson-Fisher	92,83	0,9944	0,0056	0,75575
Robinson-Fisher č. 1	93,72	0,9947	0,0053	0,6635
Robinson-Fisher č. 2	93,45	0,9948	0,0052	0,6111

Tabulka č. 28: Porovnání filtrů a kombinací experimentů na jinak rozdělených složkách při testu na sadě TREC05

5.4.9. Experiment změny typy učení

Jako další experiment je vyzkoušení, zda nejúspěšnější verze dokážou zlepšit detekci na lepší úroveň u učení se jen při chybě, než klasické učení se ze všeho, bez modifikací. Teoreticky by učení jen z chyb mělo zhoršit detekci, jelikož nedodá mnoho dalších informací do datasetu. Otázkou zůstává, zda úbytek informací bude mít nějaký zásadní vliv na zlepšování detekce aplikovaných experimentů.

Nejdříve však musím porovnat, jak moc se zhorší detekce při změně učení. Jak jsem předpokládal, výsledky v porovnání s učením se vždy jsou horší, nicméně ne až tak špatné, *viz. tabulka č. 29*. Zhoršení u vážené přesnosti je oproti zlepšovacím experimentům menší, také 1-ROCA nesnížila svou hodnotu rapidně.

V porovnání klasického nastavení a učení při chybě s využitím zlepšovacích experimentů bude záležet na tom, jak moc prostoru dostanou experimenty na zlepšení, zda to málo, kdy se použijí

bude stačit. V *tabulce č. 30* vidíme, že patrně jiný typ učení neměl viditelný špatný vliv na zlepšovací experimenty a jeho hodnoty jsou maximálně sníženy o změnu typu učení.

Filtry a změna statistik	Acc	W Acc 1/50	1-ROCA
Bayesovská kombinace	-0,45	-0,0004	-0,171
Robinsonův geom. Průměr	-0,51	-0,0003	-0,044
Robinson-Fisher	-0,20	-0,0007	-0,043

Tabulka č. 29: Změna výsledků po využití učení se jen při chybě při testu na sadě TREC05.

Filtr	Acc	W Err 1/50	1-ROCA
Bayesovská kombinace č. 1	3,04	0,0016	0,3674
Bayesovská kombinace č. 2	1,95	0,0008	-0,2831
Robinsonův geom. Průměr č. 1	3,20	0,0019	0,0757
Robinsonův geom. Průměr č. 2	3,20	0,0018	-0,0275
Robinson-Fisher č. 1	1,04	0,0003	0,1294
Robinson-Fisher č. 2	0,76	0,0006	0,1576

Tabulka č. 30: Rozdíl výsledků učení se ze všeho v klasické podobě s učením se při chybě se zlepšovacími experimenty z *tabulky č. 24* při testu na sadě TREC05

5.5. Vlastní experimenty na malé sadě

Tato podkapitola se bude věnovat testování předchozích výsledků, avšak na mnohem menší sadě emailů. Přibudou však i nové testy díky využití typu sady s celými emaily, tedy ne pouze texty.

5.5.1. Klasická účinnost filtrů na malé sadě

Abychom mohli lépe porovnávat, je třeba nejdříve otestovat, jak na tom jsou filtry v klasické podobě na této sadě. Testovat se bude ve dvou variantách datasetu a to na vyexportovaných textech z těla emailu a na celkových emailech.

Filtr	Acc	W Acc 1/50	W Err 1/50	1-ROCA
Bayesovská kombinace	98,91	0,9972	0,0028	1,3768
Robinsonův geom. Průměr	98,44	0,9980	0,0020	0,15315
Robinson-Fisher	98,88	0,9963	0,0037	0,1623

Tabulka č. 31: Výsledky filtrů při testu na malé sadě spamAssassin

Filtr	Acc	W Acc 1/50	W Err 1/50	1-ROCA
Bayesovská kombinace	97,72	0,9945	0,0055	2,01995
Robinsonův geom. Průměr	97,37	0,9966	0,0034	0,5495
Robinson-Fisher	96,99	0,9974	0,0026	0,59595

Tabulka č. 32: Výsledky filtrů při testu na malé sadě spamAssassin s celými emaily

Při pohledu na *tabulku č. 31* vidíme, že rozdíl oproti předchozí sadě je velký. V přesnosti se liší výsledky o 6% a v případě vážené přesnosti je chyba dvounásobně menší. Hodnota 1-ROCA je razantně menší. Pokud výsledky srovnáme s *tabulkou 32.*, kde se zpracovávaly celé emaily, všimneme si, že přesnost se zhoršila jen o 1 procento, avšak vážená přesnost i hodnota 1-ROCA se v prvních 2 filtrech zhoršily razantněji. Výjimkou je Robinson-Fisher, kde se vážená přesnost zlepšila nicméně 1-ROCA dosáhla razantního zhoršení. Z tohoto pohledu nevypadá, že by

zpracování celého emailu přineslo závratné cenné zlepšení, za předpokladu, že se kvůli tomuto kroku velice zvětší velikost historického datasetu.

5.5.2. Experiment sledování předmětu emailu

Pro tento experiment potřebujeme, aby byl proveden na sadě plnohodnotných emailů. Půjde nám o to, vyhodnocovat slova z předmětu separátně od slov ostatních.

Filtry a změna statistik	Acc	W Acc 1/50	1-ROCA
Bayesovská kombinace	0,08	0,0000	0,102
Robinsonův geom. Průměr	0,11	0,0002	0,025
Robinson-Fisher	0,08	-0,0001	0,060

Tabulka č. 33: Rozdíl výsledků filtrů po aplikaci experimentu sledování předmětu při testu na malé sadě spamAssassin

Při pohledu na *tabulku č. 33* lze poznat, že experiment nedosáhl žádných výrazných rozdílů. Jedna z možností, proč tomu tak je, by mohla být nalezena v množství emailů v sadě. Slova v předmětu jsou nové elementy v datasetu s vlastní statistikou, a pokud máme méně emailů, nemusí se mnoho slov stát použitelnými pro významná slova. Dalším faktorem může být již dost skvělá klasifikace filtrem, kde není již moc co vylepšovat.

5.5.3. Experiment úspěšných experimentů na velké sadě převedeno na malou

Zde využijeme předchozí uskutečněné experimenty na velké sadě a vyzkoušíme, jakou úspěšností budou disponovat. Opět použijeme nastavení z *tabulky č. 24*, kde máme pro každý filtr 2 variace. Testovat budeme nad textovou variantou malé sady.

Filtr	Acc	W Err 1/50	1-ROCA
Bayesovská kombinace č. 1	-0,13	0,0028	-2,1615
Bayesovská kombinace č. 2	-1,05	0,0023	-3,1650
Robinsonův geom. Průměr č. 1	-1,02	0,0009	-0,4281
Robinsonův geom. Průměr č. 2	-1,07	0,0010	-0,5048
Robinson-Fisher č. 1	0,58	-0,0015	-0,1769
Robinson-Fisher č. 2	0,73	-0,0040	-0,1659

Tabulka č. 34: Rozdíl výsledků klasické podoby filtru a po aplikování zlepšovacích experimentů z *tabulky č. 24* na malé sadě spamAssassin

Výsledky, které vidíme v *tabulce č. 34*, ukazují, že v přesnosti u Bayesovské kombinace a Robinsonového průměru došlo ke zhoršení detekce, nicméně vážená přesnost dopadla lépe, u Bayesovské kombinace mnohem lépe. Může za to lepší detekce hamů za cenu pár spamů. Nicméně 1-ROCA hodnota všude zhoršila svůj výsledek a to hlavně velice u Bayesovské kombinace. Robinson-Fisher inversní chí-kvadrát dosáhnul zlepšení v přesnosti, nicméně u vážené přesnosti, která je pro nás důležitější, dosahuje velkého zhoršení.

Nastává zajímavý okamžik, kdy je těžké odhadnout, co může za rozdíly v detekci. Velmi pravděpodobně to však bude souviset s rozdílnými sadami emailů a také množstvím emailů v nich.

6. Závěr

Všechny 3 zmiňované filtry se mi podařily implementovat, včetně všech plánovaných experimentálních úprav. Při srovnávaní filtrů jsem u každého z nich prozkoumal jejich rozkládaní emailů na ose a dále použil 3 statistické možnosti srovnání.

V jejich základním nastavení se z pohledu přesnosti jevila jako nejlepší Bayesovská kombinace, která dosahovala nejlepšího výsledku jak na velké, tak i na malé sadě emailů. Vážená přesnost, která vyhodnocuje poměrově, byla na velké sadě nejlepší u Bayesovské kombinace, na malé však u Robinsonova geometrického průměru. Hodnota 1-ROCA, zkoumající rozložení hodnot emailů na ose, byla nejlepší na obou sadách opět u Robinsonova geometrického průměru. Bayesovská kombinace, mající v ostatních statistikách dobré výsledky, zde dosahovala nejhorších hodnot.

Při experimentování jsem zjistil, že experimenty, jako citlivost filtru na velká a malá písmena, odstranění vykřičníků a otazníků z oddělovačů, sledování url a emailových adres, se jeví jako přínosné pro implementaci do filtrů. Experiment se sousedícími slovy našel uplatnění pouze u Bayesovské kombinace a Robinsonového geometrického průměru. Dále bylo zjištěno, že pokud se filtr učí pouze při špatné detekci, zhorší se výsledky filtru. Nicméně toto nemá nežádoucí vliv na kladně působící experimentální úpravy, které zlepšují detekci o minimálně stejný díl.

Při testu na malé sadě se projevil záporný účinek experimentálních úprav na Robinson-Fisher inversním chí-kvadrátu. Dokazuje to, že i množství emailů, nad kterými se pracuje, může ovlivňovat výsledky experimentů.

Dále bylo zjištěno, že jiné rozložení emailů ve složkách dokáže změnit celkové výsledky všech statistik. To svědčí o horších možnostech opravdu relevantního srovnávání filtrů mezi sebou. Také z tohoto důvodu nelze výsledky naměřené srovnat s výsledky testování jiných osob.

S diplomovou prací by se mohlo dále pracovat a rozvíjet ji. Lze jednoduše přidávat další filtry založené na učení, či doimplementovat experimentální funkce. Další možnost můžeme vidět v rozšíření aplikace o další statistické údaje a výpočty, které by mohly lépe pomoc porovnávat filtry mezi sebou.

V této diplomové práci jsem si vyzkoušel aplikování matematických vzorců do programu a vyzkoušel si, jak spamové filtry založené na učení fungují. Dalším kladem pro mne je zabrání se do problému statistického vyhodnocování.

7. Reference

[1] The Economist. *The etiquette of telecommunications: Getting the message, at last*. [online], publikováno 13.12.2007 [citováno 25.4.2013]. Dostupné z: <http://www.economist.com/node/10286400/print?story_id=10286400 >

[2] The SpamAssassin. *sa-learn - train SpamAssassin's Bayesian* classifier [online], publikováno 18.9.2010 [citováno 26.4.2013]. Dostupné z: <http://manpages.ubuntu.com/manpages/gutsy/man1/sa-learn.1p.html>

[3] Jonathan A. ZDZIARSKI. *Ending Spam: Bayesian Content Filtering and the Art of Statistical Language Classification.* No Starch Press 2005. ISBN:1593270526

[4] GRAHAM Paul. *A Plan for spam* [online], publikováno 1.8.2002 [citováno 22.4.2013]. Dostupné z: <http://www.paulgraham.com/spam.html>

[5] ROBINSON Gary. *Why chi?: motivations for the Use of Fisher's Inverse Chi-Square Procedure in Spam Classification,* Version 93 [online], publikováno 6.5.2004 [citováno 22.4.2013]. Dostupné z: <http://garyrob.blogs.com/whychi93.pdf >

[6] JANÍK Michal. *Stop slova* [online], publikováno 6.3.2009 [citováno 20.4.2013]. Dostupné z: <http://www.michaljanik.cz/oblibene/stop-slova>

[7] M.VOORHEES Ellen. *Overview of TREC 2005*, National Institute of Standards and Technology, Gaithersburg MD 20899

[8] SpamAssassin. *Read me file with public corpus* [online], publikováno 31.1.2006 [citováno 26.4.2013]. Dostupné z: <http://spamassassin.apache.org/publiccorpus/readme.html>

[9] SORVIK Torgeir. *Improving Spam Filtering by Training on Artificially Generated Text*. 2004. Diplomová práce fakulty počítačových věd na univerzitě Bristol, vedoucí práce Tim Kovacs

[10] FAWCETT Tom. *ROC Graphs: Notes and Practical Considerations for Data Mining Researchers* [online], publikováno 7.1.2003 [citováno 23.4.2013]. Dostupné z: <http://www.hpl.hp.com/techreports/2003/HPL-2003-4.pdf>

A Obsah CD

+ Diplomová práce	- složka obsahující všechen obsah
+ Aplikace	- složka obsahující aplikaci v solution VS 2010
- dp-prace.pdf	- text diplomové práce

B Příručka programátora

Pro správnou funkčnost je třeba správně nastavit proměnné v souboru *SettingLoader.cs*

Je zde několik cest a názvů pro testování jednoho historického datasetu a několika složek na něm (varianta 1), nebo křížového testu (varianta 2):

```
string path_default;          //cesta do složky kde se pracuje
string path_test;             //složka pro jednotlivý test(pouze varianta 1)
string path_historic;         //název pro soubor s historickým datasetem
string path_output;           //název složky pro výstupní data
string path_mails;            //název složky pro emaily datasetu(pouze varianta 1)
string[] path_tmails;         //názvy složek pro emaily na test(pouze varianta 1)
string path_histogram;        //název složky pro výstup třídy Histogram
string path_generated;        //název složky pro vygenerované složky emailů(pouze
                                varianta 2)
string path_generated_linear;//název složky pro vygenerované složky emailů stylem
                                lineárního rozdělení(pouze varianta 2)
string path_generated_cikcak;//název složky pro vygenerované složky emailů modulem
                                (pouze varianta 2)
```

Dále je zde nastavování typů experimentů apod.:

```
bool type_header;             //true = aktivní experiment s předmětem emailu
bool type_sensitive;          //true = aktivní experiment s velkými a malými písmeny
bool type_url;                //true = aktivní experiment s url a emailovými adresami
bool type_shout;              //true = aktivní experiment s vykřičníky a otazníky
bool type_numbers;            //true = aktivní experiment s čísly v oddělovačích
bool type_megaspam;           //true = aktivní experiment s megaspamem
bool type_nextwords;          //true = aktivní experiment se sousedícími slovy
bool type_couples;            //true = ten samý exp., akorát nezáleží na pořadí
bool type_fisher;             //true = aktivní experiment s R-F rozsahem
bool type_generated;          //true = aktivní test na generovaných složkách
bool type_generated_cikcak;   //true = aktivní test na generovaných složkách vytvořené
                                modulem
```

Nastavení obsahuje i další proměnné:

```
bool type_dataset;            //true = existuje již dataset, který se jen načte
                                (pouze varianta 1)
bool type_nosave;             //true = po zpracování historického datasetu se neuloží a
                                ihned se použije
int version;                  //označuje číslo verze testování
int top_size;                 //označuje velikost počtu nebo rozsahu významných slov,
                                pokud se jedná o rozsah, např. číslo 10 značí
                                vzdálenost od kraje o 0,01
int dotting;                  //tečkování během zpracování u testování
int dotting_set;              //tečkování během zpracování u tvoření datasetu
double spam_edge;             //označuje, kde se nalézá klasifikační práh filtru
int modif;                    //hodnota, která vynásobí hodnotu četnosti hamu datasetu
int megaspamEdge;             //parametr experimentu megaspam, určí, po jaké četnosti v
                                emailu je slovo kandidatem na megaspam
int megaspamKoef;             //parametr experimentu megaspam, určí, kolikrát je spam
                                přidán do významných slov
string versionName;           //informativní název verze, jež se přidá do výstupních
                                souborů
```

Posledním nastavením jsou varianty:

```
int variant_learn;           //1-> filtr se učí ze všeho
                             //2-> filtr se učí při chybě

int variant_combination;     //1-> Bayesovská kombinace
                             //2-> Robinsonův geom. průměr
                             //3-> Robinson-Fisher inversní chí-kvadrát
```

Pokud tedy chceme testovat, pravděpodobně variantu 2, musíme z cest nastavit vše, krom cest pouze pro variantu 1. Pokud chceme otestovat experimentální funkce, tak těm daným nastavíme *true*. Některé mají podmínky mezi sebou, ty jsou nastaveny tak, že není třeba je zapisovat ručně a nachází se dole v části označené "dependences".

Jako důležité je třeba nastavit variantu učení a kombinaci, reprezentující statistický filtr. Pokud bychom chtěli otestovat více možností, či testovat různé experimenty a nebo obojí, využijeme číslo verze. Toto číslo je jedním ze vstupních parametrů kostruktoru třídy, díky kterému můžeme nastavit více testování zároveň.

Ve třídě, kde vytváříme instanci nastavení vytvoříme smyčku na čísla verzí a ve třídě *SettingsLoader* dojde k nastavení požadovaných parametrů pouze pro tuto verzi. Pro každou verzi se následně uloží výstupní statistiky samostatně.

Jako ukázka využití je např. test klasického nastavení všech implementovaných filtrů:

```
if (version == 1)
{
    versionName = "Bayes";
    spam_edge = 0.9;
    variant_combination = 1;
    top_size = 15;
 }
 if (version == 2)
 {
    versionName = "Robinson ";
    versionName += "";
    spam_edge = 0.55;
    variant_combination = 2;
    top_size = 15;
  }
  if (version == 3)
  {
    versionName = "Robinson Fisher ";
    versionName += "";
    spam_edge = 0.55;
    variant_combination = 3;
    top_size = 10;
   }
```

Do podmínek však stačí vkládat parametry, které jsou rozdílné. Ty stejné můžeme definovat jednou mimo podmínky.

Třída má ještě 2 vstupní parametry, které lze využít. *Edge* a *Top*, které nastavují práh filtru a počet významných slov. Jsou zde proto, že je jednodušší čísla nastavovat smyčkou než pracně vypisovat.

Další důležitou částí pro nastavení je třída *FilesControler*. Pokud vytvoříme instanci této třídy, vytvoří nám složky s emaily rozdělenými rovnoměrně do nich podle dvou metod popsaných výše. Jako vstupní parametr je nastavení ze *SettingsLoader*, kde se využívá cest a řetězec označující složku, ve které jsou přichystány emaily na roztřízení. Podle toho, zda se jedná o vyextrahované texty nebo kompletní emaily necháme odkomentovanou metodu k tomuto typu určenou. Aplikace pak udělá vše za nás a výsledkem jsou nové složky s emaily v kořenové složce.